Geodesic Methods in Computer Vision and Graphics

Gabriel Peyré, Mickaël Péchaud, Renaud Keriven, Laurent D. Cohen

Foundations and Trends® in
sample
Vol. xx, No xx (xxxx) 1–201

DOI: xxxxxx

now
the essence of knowledge

Geodesic Methods in Computer Vision and Graphics

Gabriel Peyré[1], Mickael Péchaud[2], Renaud Keriven[3] and Laurent D. Cohen[4]

[1] *Ceremade, UMR CNRS 7534, Université Paris-Dauphine, Place de Lattre de Tassigny, 75775 Paris Cedex 16, France, peyre@ceremade.dauphine.fr*
[2] *DI, École Normale Supérieure, 45 rue d'Ulm, 75005 Paris, France, mickael.pechaud@normalesup.org*
[3] *IMAGINE-LIGM, École des Ponts ParisTech, 6 av Blaise Pascal, 77455 Marne-la-Valle, France, keriven@imagine.enpc.fr*
[4] *Ceremade, UMR CNRS 7534, Université Paris-Dauphine, Place de Lattre de Tassigny, 75775 Paris Cedex 16, France, cohen@ceremade.dauphine.fr*

Abstract

This paper reviews both the theory and practice of the numerical computation of geodesic distances on Riemannian manifolds . The notion of Riemannian manifold allows one to define a local metric (a symmetric positive tensor field) that encodes the information about the problem one wishes to solve. This takes into account a local isotropic cost (whether some point should be avoided or not) and a local anisotropy (which direction should be preferred). Using this local tensor field, the geodesic distance is used to solve many problems of practical interest such as segmentation using geodesic balls and Voronoi regions, sampling points at regular geodesic distance or meshing a domain with geodesic Delaunay triangles. The shortest paths for this Riemannian distance, the so-called geodesics, are also important because they follow

salient curvilinear structures in the domain. We show several applications of the numerical computation of geodesic distances and shortest paths to problems in surface and shape processing, in particular segmentation, sampling, meshing and comparison of shapes.
All the figures from this review paper can be reproduced by following the Numerical Tours of Signal Processing

`http://www.ceremade.dauphine.fr/~peyre/numerical-tour/`

Several textbooks exist that include description of several manifold methods for image processing, shape and surface representation and computer graphics. In particular, the reader should refer to [255, 40, 146, 213, 208, 209] for fascinating applications of these methods to many important problems in vision and graphics. This review paper is intended to give an updated tour of both foundations and trends in the area of geodesic methods in vision and graphics.

Contents

1

Theoretical Foundations of Geodesic Methods

This section introduces the notion of Riemannian manifold that is a unifying setting for all the problems considered in this review paper. This notion requires only the design of a local metric, which is then integrated over the whole domain to obtain a distance between pairs of points. The main property of this distance is that it satisfies a non-linear partial differential equation, which is at the heart of the fast numerical schemes considered in Chapter 2.

1.1 Two Examples of Riemannian Manifolds

To give a flavor of Riemannian manifolds and geodesic paths, we give two important examples in computer vision and graphics.

1.1.1 Tracking Roads in Satellite Image

An important and seminal problem in computer vision consists in detecting salient curves in images, see for instance [55]. They can be used to perform segmentation of the image, or track features. A representative example of this problem is the detection of roads in satellite images.

Figure 1.1, upper left, displays an example of satellite image f, that is modeled as a 2D function $f : \Omega \to \mathbb{R}$, where the image domain is usually $\Omega = [0,1]^2$. A simple model of road is that it should be approximately of constant gray value $c \in \mathbb{R}$. One can thus build a saliency map $W(x)$ that is low in area where there is a high confidence that some road is passing by, as suggested for instance in [68]. As an example, one can define

$$W(x) = |f(x) - c| + \varepsilon \tag{1.1}$$

where ε is a small value that prevents $W(x)$ from vanishing.

Using this saliency map, one defines the length of a smooth curve on the image $\gamma : [0,1] \to \Omega$ as a weighted length

$$L(\gamma) = \int_0^1 W(\gamma(t)) \|\gamma'(t)\| \mathrm{d}t \tag{1.2}$$

where $\gamma'(t) \in \mathbb{R}^2$ is the derivative of γ. We note that this measure of lengths extends to piecewise smooth curves by splitting the integration into pieces where the curve is smooth.

The length $L(\gamma)$ is smaller when the curve passes by regions where W is small. It thus makes sense to declare as roads the curves that minimize $L(\gamma)$. For this problem to make sense, one needs to further constrain γ. And a natural choice is to fix its starting and ending points to be a pair $(x_s, x_e) \in \Omega^2$

$$\mathcal{P}(x_s, x_e) = \{\gamma : [0,1] \to \Omega \setminus \gamma(0) = x_s \quad \text{and} \quad \gamma(1) = x_e\}, \tag{1.3}$$

where the paths are assumed to be piecewise smooth so that one can measure their lengths using (1.2).

Within this setting, a road $\gamma^\star$ is a global minimizer of the length

$$\gamma^\star = \operatorname*{argmin}_{\gamma \in \mathcal{P}(x_s, x_e)} L(\gamma), \tag{1.4}$$

which in general exists, and is unique except in degenerate situations where different roads have the same length. Length $L(\gamma^\star)$ is called geodesic distance between x_s and x_e with respect to $W(x)$.

Figure 1.1 shows an example of geodesic extracted with this method. It links two points x_s and x_e given by the user. One can see that this

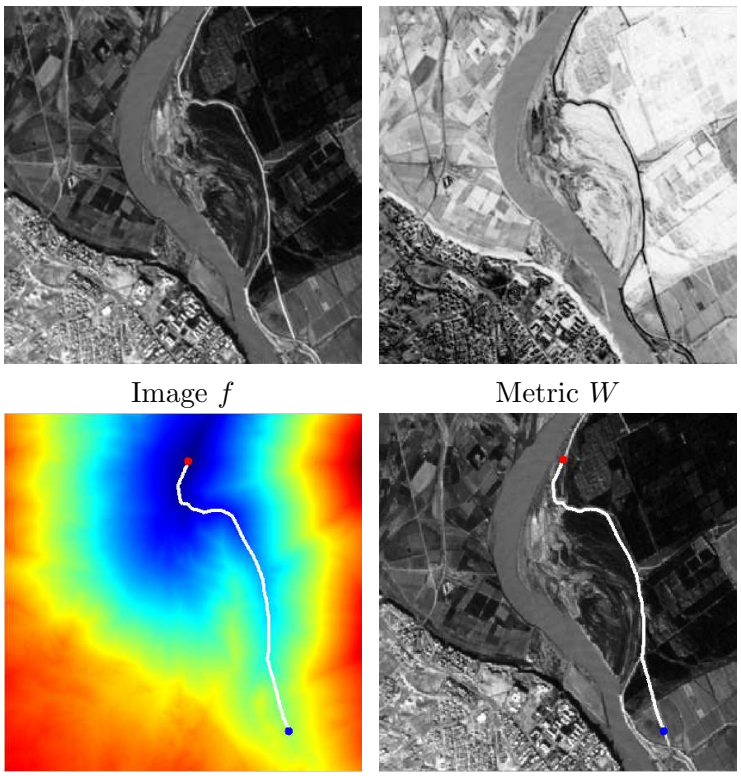


Fig. 1.1 *Example of geodesic curve extracted using the weighted metric* (1.1). x_s *and* x_e *correspond respectively to the red and blue points*

curve tends to follow regions with gray values close to c, which has been fixed to $c = f(x_e)$.

This idea of using a scalar potential $W(x)$ to weight the length of curves has been used in many computer vision applications beside road tracking. This includes in particular medical imaging where one wants to extract contours of organs or blood vessels. These applications are further detailed in Chapter 3.

1.1.2 Detecting Salient Features on Surfaces

Computer graphics applications often face problems that require the extraction of meaningful curves on surfaces. We consider here a

smooth surface $\mathcal{S}$ embedded into the 3D Euclidean space, $\mathcal{S} \subset \mathbb{R}^3$.

Similarly to (1.2), a curve $\tilde{\gamma} : [0,1] \to \mathcal{S}$ traced on the surface has a weighted length computed as

$$L(\tilde{\gamma}) = \int_0^1 W(\tilde{\gamma}(t)) \|\tilde{\gamma}'(t)\| \mathrm{d}t, \tag{1.5}$$

where $\tilde{\gamma}'(t) \in \mathcal{T}_{\tilde{\gamma}(t)} \subset \mathbb{R}^3$ is the derivative vector, that lies in the embedding space $\mathbb{R}^3$, and is in fact a vector belonging to the 2D tangent plane $\mathcal{T}_{\tilde{\gamma}(t)}$ to the surface at $\tilde{\gamma}(t)$, and the weight W is a positive function defined on the surface domain $\mathcal{S}$.

Note that we use the notation $\tilde{x} = \tilde{\gamma}(t)$ to insist on the fact that the curves are not defined in a Euclidean space, and are forced to be traced on a surface.

Similarly to (1.4), a geodesic curve

$$\tilde{\gamma}^\star = \underset{\tilde{\gamma} \in \mathcal{P}(\tilde{x}_s, \tilde{x}_e)}{\operatorname{argmin}} \; L(\tilde{\gamma}), \tag{1.6}$$

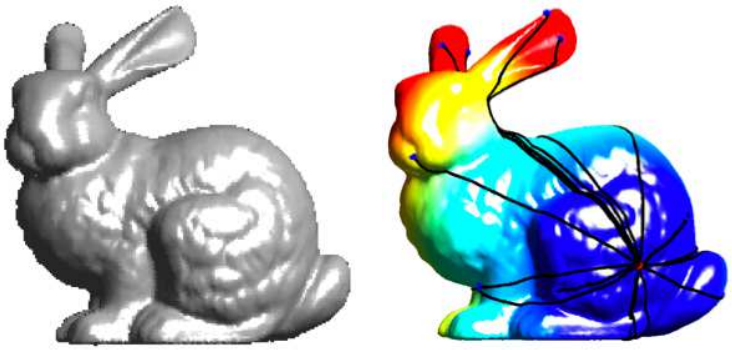
is a shortest curve joining two points $\tilde{x}_s, \tilde{x}_e \in \mathcal{S}$.

When $W = 1$, $L(\tilde{\gamma})$ is simply the length of a 3D curve, that is restricted to be on the surface $\mathcal{S}$. Figure 1.2 shows an example of surface, together with a set of geodesics joining pairs of points, for $W = 1$. As detailed in Section 3.2.4, a varying saliency map $W(\tilde{x})$ can be defined from a texture or from the curvature of the surface to detect salient curves.

Geodesics and geodesic distance on 3D surfaces have found many applications in computer vision and graphics, for example surface matching, detailed in Chapter 5, and surface remeshing, detailed in Chapter 4.

1.2 Riemannian Manifolds

It turns out that both previous examples can be cast into the same general framework using the notion of a Riemannian manifold of dimension 2.



Surface $\mathcal{S}$ Geodesic distance and paths.

Fig. 1.2 *Example of geodesic curves on a 3D surface.*

1.2.1 Surfaces as Riemannian Manifolds

Although the curves described in sections 1.1.1 and 1.1.2 do not belong to the same spaces, it is possible to formalize the computation of geodesics in the same way in both cases. In order to do so, one needs to introduce the Riemannian manifold $\Omega \subset \mathbb{R}^2$ associated to the surface $\mathcal{S}$ [148].

A smooth surface $\mathcal{S} \subset \mathbb{R}^3$ can be locally described as a parametric function

$$\varphi : \begin{array}{ccc} \Omega \subset \mathbb{R}^2 & \rightarrow & \mathcal{S} \subset \mathbb{R}^3 \\ x & \mapsto & \tilde{x} = \varphi(x) \end{array} \tag{1.7}$$

which is required to be differentiable and one-to-one, where Ω is an open domain of $\mathbb{R}^2$.

Full surfaces require several such mappings to be fully described, but we postpone this difficulty until Section 1.2.2.

The tangent plane $\mathcal{T}_{\tilde{x}}$ at a surface point $\tilde{x} = \varphi(x)$ is spanned by the two partial derivatives of the parameterization, which define the derivative matrix at point $x = (x_1, x_2)$

$$D\varphi(x) = \left(\frac{\partial \varphi}{\partial x_1}(x), \frac{\partial \varphi}{\partial x_2}(x) \right) \in \mathbb{R}^{3 \times 2}. \tag{1.8}$$

As shown in Figure 1.3, the derivative of any curve $\tilde{\gamma}$ at a point $\tilde{x} = \tilde{\gamma}(t)$ belongs to the tangent plane $\mathcal{T}_{\tilde{x}}$ of $\mathcal{S}$ at $\tilde{x}$.

The curve $\tilde{\gamma}(t) \in \mathcal{S} \subset \mathbb{R}^3$ defines a curve $\gamma(t) = \varphi^{-1}(\tilde{\gamma}(t)) \in \Omega$

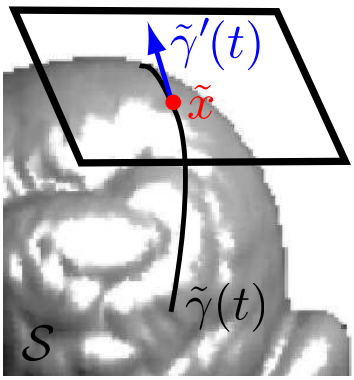


Fig. 1.3 *Tangent space $\mathcal{T}_{\tilde{x}}$ and derivative of a curve on surface $\mathcal{S}$.*

traced on the parameter domain. Note that while $\tilde{\gamma}$ belongs to a curved surface, γ is traced on a subset of a Euclidean domain.

Since $\tilde{\gamma}(t) = \varphi(\gamma(t)) \in \Omega$ the tangents to the curves are related via $\tilde{\gamma}'(t) = D\varphi(\gamma(t))\gamma'(t)$ and $\tilde{\gamma}'(t)$ is in the tangent plane $\mathcal{T}_{\tilde{\gamma}(t)}$ which is spanned by the columns of $D\varphi(\gamma(t))$. The length (1.5) of the curve $\tilde{\gamma}$ is computed as

$$L(\tilde{\gamma}) = L(\gamma) = \int_0^1 \|\gamma'(t)\|_{T_{\gamma(t)}} \mathrm{d}t, \tag{1.9}$$

where the tensor T_x is defined as

$$\forall\, x \in \Omega, \quad T_x = \sqrt{W(\tilde{x})} I_\varphi(x) \quad \text{where} \quad \tilde{x} = \varphi(x),$$

and $I_\varphi(x) \in \mathbb{R}^{2\times 2}$ is the first fundamental form of $\mathcal{S}$

$$I_\varphi(x) = (D\varphi(x))^{\mathrm{T}} D\varphi(x) = \left(\langle \frac{\partial \varphi}{\partial x_i}(x), \frac{\partial \varphi}{\partial x_j}(x) \rangle \right)_{1 \leqslant i,j \leqslant 2} \tag{1.10}$$

and where, given some positive symmetric matrix $A = (A_{i,j})_{1\leqslant i,j\leqslant 2} \in \mathbb{R}^{2\times 2}$, we define its associated norm

$$\|u\|_A^2 = \langle u, u \rangle_A \quad \text{where} \quad \langle u, v \rangle_A = \langle Au, v \rangle = \sum_{1 \leqslant i,j \leqslant 2} A_{i,j} u_i v_j. \tag{1.11}$$

A domain Ω equipped with such a metric is called a Riemannian manifold.

The geodesic curve $\tilde{\gamma}^\star$ traced on the surface $\mathcal{S}$ defined in (1.6) can equivalently be viewed as a geodesic $\gamma^\star = \varphi^{-1}(\tilde{\gamma}^\star)$ traced on the Riemannian manifold Ω. While $\tilde{\gamma}^\star$ minimizes the length (1.5) in the 3D embedding space between $\tilde{x}_s$ and $\tilde{x}_e$ the curve $\gamma^\star$ minimizes the Riemannian length (1.9) between $x_s = \varphi^{-1}(\tilde{x}_s)$ and $x_e = \varphi^{-1}(\tilde{x}_e)$.

1.2.2 Riemannian Manifold of Arbitrary Dimensions

Local description of a manifold without boundary. We consider an arbitrary manifold $\mathcal{S}$ of dimension d embedded in $\mathbb{R}^n$ for some $n \geqslant d$ [164]. This generalizes the setting of the previous Section 1.2.1 that considers $d = 2$ and $n = 3$. The manifold is assumed for now to be closed, which means without boundary.

As already done in (1.7), the manifold is described locally using a bijective smooth parametrization

$$\varphi : \begin{array}{ccc} \Omega \subset \mathbb{R}^d & \to & \mathcal{S} \subset \mathbb{R}^n \\ x & \mapsto & \tilde{x} = \varphi(x) \end{array}$$

so that $\varphi(\Omega)$ is an open subset of $\mathcal{S}$.

All the objects we consider, such as curves and length, can be transposed from $\mathcal{S}$ to Ω using this application. We can thus restrict our attention to Ω, and do not make any reference to the surface $\mathcal{S}$.

For an arbitrary dimension d, a Riemannian manifold is thus locally described as a subset of the ambient space $\Omega \subset \mathbb{R}^d$, having the topology of an open sphere, equipped with a positive definite matrix $T_x \in \mathbb{R}^{d \times d}$ for each point $x \in \Omega$, that we call a tensor field. This field is further required to be smooth.

Similarly to (1.11), at each point $x \in \Omega$, the tensor T_x defines the length of a vector $u \in \mathbb{R}^d$ using

$$\|u\|_{T_x}^2 = \langle u, u \rangle_{T_x} \quad \text{where} \quad \langle u, v \rangle_{T_x} = \langle T_x u, v \rangle = \sum_{1 \leqslant i,j \leqslant d} (T_x)_{i,j} u_i v_j.$$

This allows one to compute the length of a curve $\gamma(t) \in \Omega$ traced on the Riemannian manifold as a weighted length where the infinitesimal length is measured according to T_x

$$L(\gamma) = \int_0^1 \|\gamma'(t)\|_{T_{\gamma(t)}} \mathrm{d}t. \tag{1.12}$$

The weighted metric on the image for road detection defined in Section 1.1.1 fits within this framework for $d = 2$ by considering $\Omega = [0,1]^2$ and $T_x = W(x)^2 \mathrm{Id}_2$, where $\mathrm{Id}_2 \in \mathbb{R}^{2\times 2}$ is the identity matrix. In this case, $\Omega = \mathcal{S}$, and φ is the identity application. The parameter domain metric defined from a surface $\mathcal{S} \subset \mathbb{R}^3$ considered in Section 1.1.2 can also be viewed as a Riemannian metric as we explained in the previous section.

Global description of a manifold without boundary. The local description of the manifold as a subset $\Omega \subset \mathbb{R}^d$ of an Euclidean space is only able to describe parts that are topologically equivalent to open spheres.

A manifold $\mathcal{S} \in \mathbb{R}^n$ embedded in $\mathbb{R}^n$ with an arbitrary topology is decomposed using a finite set of overlapping surfaces $\{\mathcal{S}_i\}_i$ topologically equivalent to open spheres such that

$$\bigcup_i \mathcal{S}_i = \mathcal{S}. \tag{1.13}$$

A chart $\varphi_i : \{\Omega_i\}_i \to \mathcal{S}_i$ is defined for each of sub-surface $\mathcal{S}_i$.

Figure 1.4 shows how a 1D circle is locally parameterized using several 1D segments.

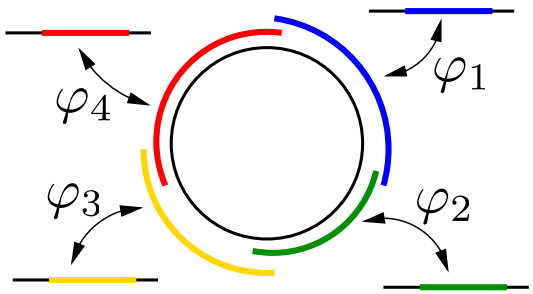


Fig. 1.4 *The circle is a 1-dimensional surface embedded in $\mathbb{R}^2$, and is thus a 1D manifold. In this example, it is decomposed in 4 sub-surfaces which are topologically equivalent to sub-domains of $\mathbb{R}$, through charts φ_i.*

Manifolds with boundaries In applications, one often encounters manifolds with boundaries, for instance images defined on a square,

volume of data defined inside a cube, or planar shapes.

The boundary $\partial\Omega$ of a manifold Ω of dimension d is itself by definition a manifold of dimension $d-1$. Points x strictly inside the manifold are assumed to have a local neighborhood that can be parameterized over a small Euclidean ball. Points located on the boundary are parameterized over a half Euclidean ball.

Such manifolds require some extra mathematical care, since geodesic curves (local length minimizers) and shortest paths (global length minimizing curves), defined in Section 1.2.3, might exhibit tangential discontinuities when reaching the boundary of the manifold.

Note however that these curves can still be computed numerically as described in Chapter 2. Note also that the characterization of the geodesic distance as the viscosity solution of the Eikonal equation still holds for manifolds with boundary.

1.2.3 Geodesic Curves

Globally minimizing shortest paths. Similarly to (1.4), one defines a geodesic $\gamma^\star(t) \in \Omega$ between two points $(x_s, x_e) \in \Omega^2$ as the curve between x_s and x_e with minimal length according to the Riemannian metric (1.9):

$$\gamma^\star = \operatorname*{argmin}_{\gamma \in \mathcal{P}(x_s,x_e)} L(\gamma). \tag{1.14}$$

As an example, in the case of a uniform $T_x = Id_d$ (i.e. the metric is Euclidean) and a convex Ω, the unique geodesic curve between x_s and x_e is the segment joining the two points.

Existence of shortest paths between any pair of points on a connected Riemannian manifold is guaranteed by the Hopf-Rinow theorem [133]. Such a curve is not always unique, see Figure 1.5.

Locally minimizing geodesic curves. It is important to note that in this paper the notion of geodesics refers to minimal paths, that are curves minimizing globally the Riemannian length between two points. In contrast, the mathematical definition of geodesic curves usually refers to curves that are local minimizer of the geodesic lengths. These locally minimizing curves are the generalization of straight lines

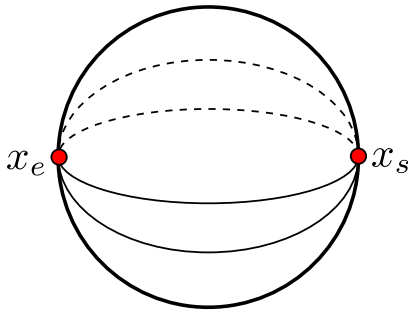


Fig. 1.5 Example of non-uniqueness of a shortest path between two points : there is an infinite number of shortest paths between two antipodal points on a sphere.

in Euclidean geometry to the setting of Riemannian manifolds.

Such a locally minimizing curve satisfies an ordinary differential equation, that expresses that it has a vanishing Riemannian curvature.

There might exist several local minimizers of the length between two points, which are not necessarily minimal paths. For instance, on a sphere, a great circle passing by two points is composed of two local minimizer of the length, and only one of the two portion of circle is a minimal path.

1.2.4 Geodesic Distance

The geodesic distance between two points x_s, x_e is the length of $\gamma^\star$.

$$d(x_s, x_e) = \min_{\gamma \in \mathcal{P}(x_s, x_e)} L(\gamma) = L(\gamma^\star). \tag{1.15}$$

This defines a metric on Ω, which means that it is symmetric $d(x_s, x_e) = d(x_e, x_s)$, that $d(x_s, x_e) > 0$ unless $x_s = x_e$ and then $d(x_s, x_e) = 0$, and that it satisfies the triangular inequality for every point y

$$d(x_s, x_e) \leqslant d(x_s, y) + d(y, x_e).$$

The minimization (1.15) is thus a way to transfer a local metric defined point-wise on the manifold Ω into a global metric that applies to arbitrary pairs of points on the manifold.

This metric $d(x_s, x_e)$ should not be mistaken for the Euclidean metric $\|x_s - x_e\|$ on $\mathbb{R}^n$, since they are in general very different. As an example, if r denotes the radius of the sphere in Figure 1.5, the Euclidean distance between two antipodal points is $2r$ while the geodesic distance is πr.

1.2.5 Anisotropy

Let us assume that Ω is of dimension 2. To analyze locally the behavior of a general anisotropic metric, the tensor field is diagonalized as

$$T_x = \lambda_1(x) e_1(x) e_1(x)^{\mathrm{T}} + \lambda_2(x) e_2(x) e_2(x)^{\mathrm{T}}, \tag{1.16}$$

where $0 < \lambda_1(x) \leqslant \lambda_2(x)$. The vector fields $e_i(x)$ are orthogonal eigenvectors of the symmetric matrix T_x with corresponding eigenvalues $\lambda_i(x)$. The norm of a tangent vector $v = \gamma'(t)$ of a curve at a point $x = \gamma(t)$ is thus measured as

$$\|v\|_{T_x} = \lambda_1(x) |\langle e_1(x),\, v\rangle|^2 + \lambda_2(x) |\langle e_2(x),\, v\rangle|^2.$$

A curve γ is thus locally shorter near x if its tangent $\gamma'(t)$ is collinear to $e_1(x)$, as shown on Figure 1.6. Geodesic curves thus tend to be as parallel as possible to the eigenvector field $e_1(x)$. This diagonalization (1.16) carries over to arbitrary dimension d by considering a family of d eigenvector fields.

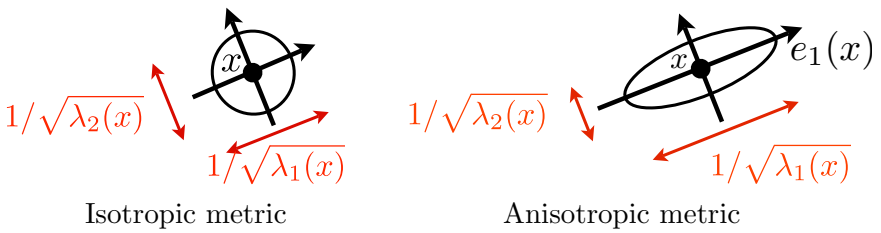


Fig. 1.6 *Schematic display of a local geodesic ball for an isotropic metric or an anisotropic metric.*

For image analysis, in order to find significant curves as geodesics of a Riemannian metric, the eigenvector field $e_1(x)$ should thus match the orientation of edges or of textures, as this is the case for Figure 1.7, right.

The strength of the directionality of the metric is measured by its anisotropy $A(x)$, while its global isotropic strength is measured using

its energy $W(x)$

$$A(x) = \frac{\lambda_2(x) - \lambda_1(x)}{\lambda_2(x) + \lambda_1(x)} \in [0,1] \quad \text{and} \quad W(x)^2 = \frac{\lambda_2(x) + \lambda_1(x)}{2} > 0. \tag{1.17}$$

A tensor field with $A(x) = 0$ is isotropic and thus verifies $T_x = W(x)^2 \mathrm{Id}_2$, which corresponds to the setting considered in the road tracking application of Section 1.1.1.

Figure 1.7 shows examples of metric with a constant energy $W(x) = W$ and an increasing anisotropy $A(x) = A$. As the anisotropy A drops to 0, the Riemannian manifold comes closer to Euclidean, and geodesic curves become line segments.

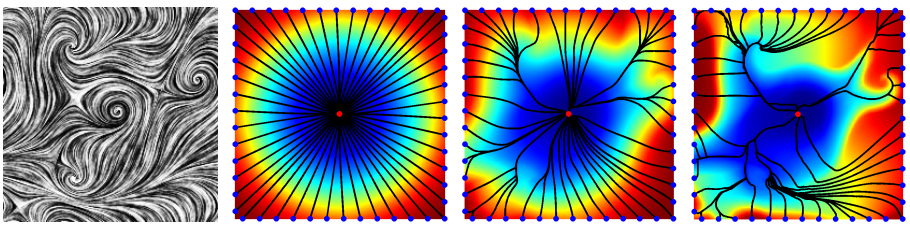


Fig. 1.7 *Example of geodesic distance to the center point, and geodesic curves between this center point and points along the boundary of the domain. These are computed for a metric with an increasing value of anisotropy A, and for a constant W. The metric is computed from the image f using* (4.37).

1.3 Other Examples of Riemannian Manifolds

One can find many occurrences of the notion of Riemannian manifold to solve various problems in computer vision and graphics. All these methods build, as a pre-processing step, a metric T_x suited for the problem to solve, and use geodesics to integrate this local distance information into globally optimal minimal paths. Figure 1.8 synthesizes different possible Riemannian manifolds. The last two columns correspond to examples already considered in sections 1.1.1 and 1.2.5.

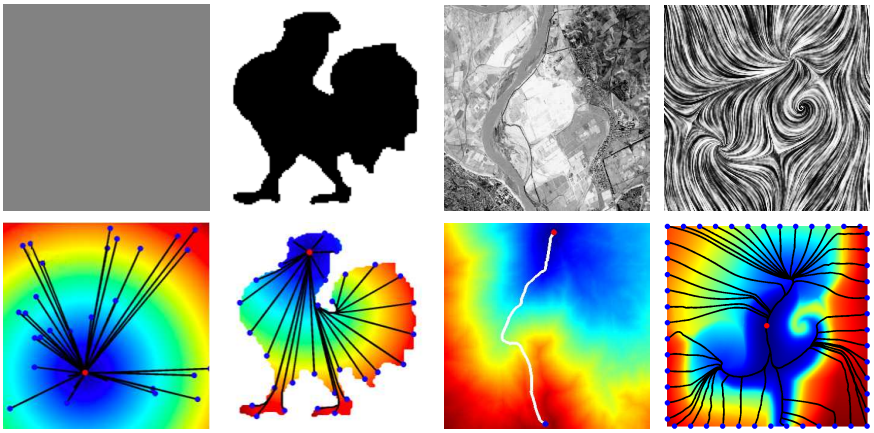


Fig. 1.8 *Examples of Riemannian metrics (top row), geodesic distances and geodesic curves (bottom row). The blue/red color-map indicates the geodesic distance to the starting red point. From left to right: Euclidean ($T_x = \mathrm{Id}_2$ restricted to $\Omega = [0,1]^2$), planar domain ($T_x = \mathrm{Id}_2$ restricted to $\mathcal{M} \subset [0,1]^2$), isotropic metric ($\Omega = [0,1]^2$, $T(x) = W(x)Id_2$, see equation (1.1)), Riemannian manifold metric (T_x is the structure tensor of the image, see equation (4.37)).*

1.3.1 Euclidean Distance

The classical Euclidean distance $d(x_s, x_e) = \|x_s - x_e\|$ in $\Omega = \mathbb{R}^d$ is recovered by using the identity tensor $T_x = \mathrm{Id}_d$. For this identity metric, shortest paths are line segments. Figure 1.8, first column, shows this simple setting. This is generalized by considering a constant metric $T_x = T \in \mathbb{R}^{2\times 2}$, in which case the Euclidean metric is measured according to T, since $d(x_s, x_e) = \|x_s - x_e\|_T$. In this setting, geodesics between two points are straight lines.

1.3.2 Planar Domains and Shapes

If one uses a locally Euclidean metric $T_x = \mathrm{Id}_2$ in 2D, but restricts the domain to a non-convex planar compact subset $\Omega \subset \mathbb{R}^2$, then the geodesic distance $d(x_s, x_e)$ might differ from the Euclidean length $\|x_s - x_e\|$. This is because paths are restricted to lie inside Ω, and some shortest paths are forced to follow the boundary of the domain, thus deviating from line segment (see Figure 1.8, second column).

This shows that the global integration of the local length measure

T_x to obtain the geodesic distance $d(x_s, x_e)$ takes into account global geometrical and topological properties of the domain. This property is useful to perform shape recognition, that requires some knowledge of the global structure of a shape $\Omega \subset \mathbb{R}^2$, as detailed in Chapter 5.

Such non-convex domain geodesic computation also found application in robotics and video games, where one wants to compute an optimal trajectory in an environment consisting of obstacles, or in which some positions are forbidden [161, 151]. This is detailed in Section 3.6.

1.3.3 Anisotropic Metric on Images

Section 1.1.1 detailed an application of geodesic curve to road tracking, where the Riemannian metric is a simple scalar weight computed from some image f. This weighting scheme does not take advantage of the local orientation of curves, since the metric $W(x)\|\gamma'(t)\|$ is only sensitive to the amplitude of the derivative.

One can improve this by computing a 2D tensor field T_x at each pixel location $x \in \mathbb{R}^{2\times 2}$. The precise definition of this tensor depends on the precise applications, see Section 3.2. They generally take into account the gradient $\nabla f(x)$ of the image f around the pixel x, to measure the local directionality of the edges or the texture. Figure 1.8, right, shows an example of metric designed to match the structure of a texture.

1.4 Voronoi Segmentation and Medial Axis

1.4.1 Voronoi Segmentation

For a finite set $S = \{x_i\}_{i=0}^{K-1}$ of starting points, one defines a segmentation of the manifold Ω into Voronoi cells

$$\Omega = \bigcup_i \mathcal{C}_i \quad \text{where} \quad \mathcal{C}_i = \{x \in \Omega \setminus \forall\, j \neq i,\ d(x, x_j) \geqslant d(x, x_i)\}\,. \tag{1.18}$$

Each region $\mathcal{C}_i$ can be interpreted as a region of influence of x_i. Section 2.6.1 details how to compute this segmentation numerically, and Section 4.1.1 discusses some applications.

This segmentation can also be represented using a partition function

$$\ell(x) = \operatorname*{argmin}_{0 \leqslant i < K} d(x, x_i). \tag{1.19}$$

For points x which are equidistant from at least two different starting points x_i and x_j, i.e. $d(x, x_i) = d(x, x_j)$, one can pick either $\ell(x) = i$ or $\ell(x) = j$. Except for these exceptional points, one thus has $\ell(x) = i$ if and only if $x \in \mathcal{C}_i$.

Figure 1.9, top row, shows an example of Voronoi segmentation for an isotropic metric.

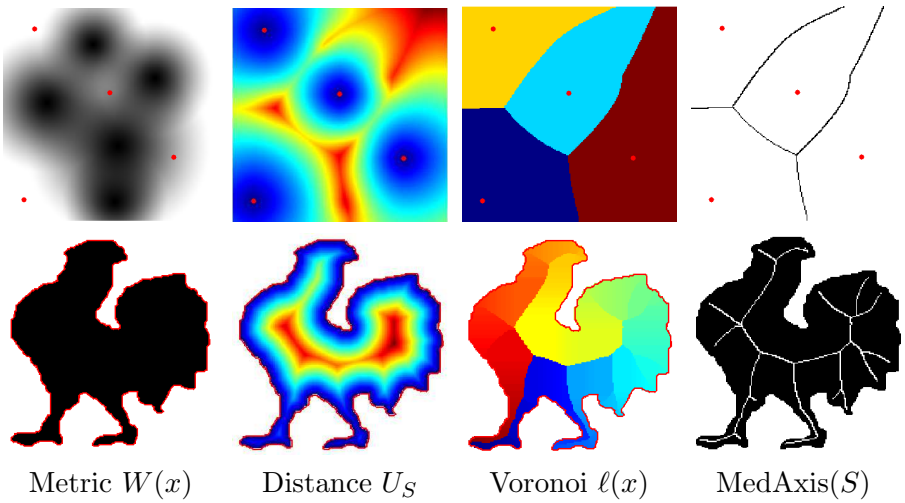


Fig. 1.9 *Examples of distance function, Voronoi segmentation and medial axis for an isotropic metric (top left) and a constant metric inside a non-convex shape (bottom left).*

This partition function $\ell(x)$ can be extended to the case where S is not a discrete set of points, but for instance the boundary of a 2D shape. In this case, $\ell(x)$ is not integer valued but rather indicates the location of the closest point in S. Figure 1.9, bottom row, shows an example for a Euclidean metric restricted to a non-convex shape, where S is the boundary of the domain. In the third image, the colors are mapped to the points of the boundary S, and the color of each point x conresponds to the one associated with $\ell(x)$.

1.4.2 Medial Axis

The medial axis is the set of points where the distance function U_S is not differentiable. This corresponds to the set of points $x \in \Omega$ where two distinct shortest paths join x to S.

The major part of the medial axis is thus composed of points that are at the same distance from two points in S

$$\left\{ x \in \Omega \setminus \exists (x_1, x_2) \in S^2 \;\middle|\; \begin{array}{l} x_1 \neq x_2 \\ d(x, x_1) = d(x, x_2) \end{array} \right\} \subset \mathrm{MedAxis}(S). \tag{1.20}$$

This inclusion might be strict because it might happen that two points $x \in \Omega$ and $y \in S$ are linked by two different geodesics.

Finite set of points. For a discrete finite set $S = \{x_i\}_{i=0}^{N-1}$, a point x belongs to $\mathrm{MedAxis}(S)$ either if it is on the boundary of a Voronoi cell, or if two distinct geodesics are joining x to a single point of S. One thus has the inclusion

$$\bigcup_{x_i \in S} \partial \mathcal{C}_i \subset \mathrm{MedAxis}(S) \tag{1.21}$$

where $\mathcal{C}_i$ is defined in (1.18).

For instance, if $S = \{x_0, x_1\}$ and if T_x is a smooth metric, then $\mathrm{MedAxis}(S)$ is a smooth mediatrix hyper surface of dimension $d-1$ between the two points. In the Euclidean case, $T_x = \mathrm{Id}_d$, it corresponds to the separating affine hyperplane.

As detailed in Section 4.1.1, for a 2D manifold and a generic dense enough configuration of points, it is the union of portion of mediatrices between pairs of points, and triple points that are equidistant from three different points of S.

Section 2.6.2 explains how to compute numerically the medial axis.

Shape skeleton. The definition (1.20) of $\mathrm{MedAxis}(S)$ still holds when S is not a discrete set of points. The special case considered in Section 1.3.2 where Ω is a compact subset of $\mathbb{R}^d$ and $S = \partial\Omega$ is of particular importance for shape and surface modeling. In this setting, $\mathrm{MedAxis}(S)$ is often called the skeleton of the shape S, and is

an important perceptual feature used to solve many computer vision problems. It has been studied extensively in computer vision as a basic tool for shape retrieval, see for instance [252]. One of the main issues is that the skeleton is very sensitive to local modifications of the shape, and tends to be complicated for non-smooth shapes.

Section 2.6.2 details how to compute and regularize numerically the skeleton of a shape. Figure 1.9 shows an example of skeleton for a 2D shape.

1.5 Geodesic Distance and Geodesic Curves

1.5.1 Geodesic Distance Map

The geodesic distance between two points defined in (1.15) can be generalized to the distance from a point x to a set of points $S \subset \Omega$ by computing the distance from x to its closest point in Ω, which defines the distance map

$$U_S(x) = \min_{y \in S} d(x, y). \tag{1.22}$$

Similarly a geodesic curve $\gamma^\star$ between a point $x \in \Omega$ and S is a curve $\gamma^\star \in \mathcal{P}(x, y)$ for some $y \in S$ such that $L(\gamma^\star) = U_S(x)$.

Figure 1.8, bottom row, shows examples of geodesic distance map to a single starting point $S = \{x_s\}$.

Figure 1.10 is a three dimensional illustration of distance maps for a Euclidean metric in $\mathbb{R}^2$ from one (left) or two (right) starting points.

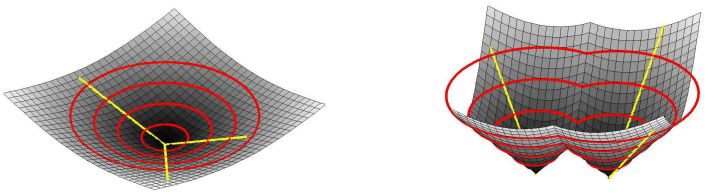
1.5.2 Eikonal Equation

For points x outside both the medial axis $\mathrm{MedAxis}(S)$ defined in (1.20) and S, one can prove that the geodesic distance map U_S is differentiable, and that it satisfies the following non-linear partial differential equation

$$\|\nabla U_S(x)\|_{T_x^{-1}} = 1, \text{ with boundary conditions } U_S(x) = 0 \text{ on } S, \tag{1.23}$$

where ∇U_S is the gradient vector of partial differentials in $\mathbb{R}^d$.

Unfortunately, even for a smooth metric T_x and simple set S, the medial axis $\mathrm{MedAxis}(S)$ is non-empty (see Figure 1.10, right, where the geodesic distance is clearly not differentiable at points equidistant



One starting point Two starting points

Fig. 1.10 *Examples of geodesic distances and curves for a Euclidean metric with different starting configurations. Geodesic distance is displayed as an elevation map over* $\Omega = [0,1]^2$. *Red curves correspond to iso-geodesic distance lines, while yellow curves are examples of geodesic curves.*

from the starting points). To define U_S as a solution of a PDE even at points where it is not differentiable, one has to resort to a notion of weak solution. For a non-linear PDE such as (1.23), the correct notion of weak solution is the notion of viscosity solution, developed by Crandall and Lions [82, 81, 80].

A continuous function u is a viscosity solution of the Eikonal equation (1.23) if and only if for any continuously differentiable mapping $\varphi \in \mathrm{C}^1(\Omega)$ and for all $x_0 \in \Omega \backslash S$ local minimum of $u - \varphi$ we have

$$\|\nabla \varphi(x_0)\|_{T_{x_0}^{-1}} = 1$$

For instance in 1D, $d = 1$, $\Omega = \mathbb{R}$, the distance function

$$u(x) = U_S(x) = \min(|x - x_1|, |x - x_2|)$$

from two points $S = \{x_1, x_2\}$ satisfies $|u'| = 1$ wherever it is differentiable. However, many other functions satisfies the same property, for example v, as shown on Figure 1.11. Figure 1.11, top, shows a $\mathrm{C}^1(\mathbb{R})$ function φ that reaches a local minimum for $u - \varphi$ at x_0. In this case, the equality $|\varphi'(x_0)| = 1$ holds. This condition would not be verified by v at point x_0. An intuitive vision of the definition of viscosity solution is that it prevents appearance of such inverted peaks outside S.

An important result from the viscosity solution of Hamilton-Jacobi equation, proved in [82, 81, 80], is that if S is a compact set, if $x \mapsto T_x$ is a continuous mapping, then the geodesic distance map U_S defined in

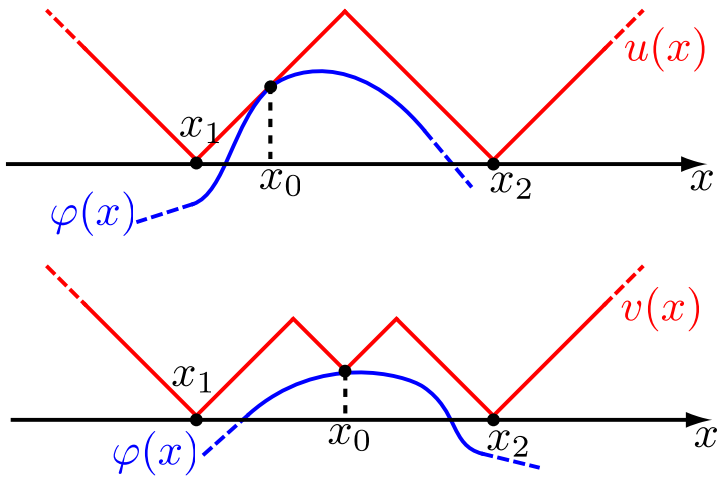


Fig. 1.11 *Schematic view in 1D of the viscosity solution constrain.*

(1.22) is the unique viscosity solution of the following Eikonal equation

$$\begin{cases} \forall\, x \in \Omega, & \|\nabla U_S(x)\|_{T_x^{-1}} = 1, \\ \forall\, x \in S, & U_S(x) = 0. \end{cases} \tag{1.24}$$

In the special case of an isotropic metric $T_x = W(x)^2 \mathrm{Id}_d$, one recovers the classical Eikonal equation

$$\forall\, x \in \Omega, \quad \|\nabla U_S(x)\| = W(x). \tag{1.25}$$

For the Euclidean case, $W(x) = 1$, one has $\|\nabla U_S(x)\| = 1$, whose viscosity solution for $S = \{x_s\}$ is $U_{x_s}(x) = \|x - x_s\|$.

1.5.3 Geodesic Curves

If the geodesic distance U_S is known, for instance by solving the Eikonal equation, a geodesic $\gamma^\star$ between some end point x_e and S is computed by gradient descent. This means that $\gamma^\star$ is the solution of the following ordinary differential equation

$$\begin{cases} \forall\, t > 0, & \dfrac{\mathrm{d}\gamma^\star(t)}{\mathrm{d}t} = -\eta_t v(\gamma^\star(t)), \\ \gamma^\star(0) = x_e. \end{cases} \tag{1.26}$$

where the tangent vector to the curve is the gradient of the distance, twisted by T_x^{-1}

$$v(x) = T_x^{-1} \nabla U_S(x),$$

and where $\eta_t > 0$ is a scalar function that controls the speed of the geodesic parameterization. To obtain a unit speed parameterization, $\|(\gamma^\star)'(t)\| = 1$, one needs to use

$$\eta_t = \|v(\gamma^\star(t))\|^{-1}.$$

If x_e is not on the medial axis $\mathrm{MedAxis}(S)$, the solution of (1.26) will not cross the medial axis for $t > 0$, so its solution is well defined for $0 \leqslant t \leqslant t_{x_e}$, for some t_{x_e} such that $\gamma^\star(t_{x_e}) \in S$.

For an isotropic metric $T_x = W(x)^2 \mathrm{Id}_d$, one recovers the gradient descent of the distance map proposed in [70]

$$\forall\, t > 0,\ \frac{\mathrm{d}\gamma^\star(t)}{\mathrm{d}t} = -\eta_t \nabla U_S(\gamma^\star(t)).$$

Figure 1.10 illustrates the case where $T_x = \mathrm{Id}_2$: geodesic curves are straight lines orthogonal to iso-geodesic distance curves, and correspond to greatest slopes curves, since the gradient of a function is always orthogonal to its level curves.

2

Numerical Foundations of Geodesic Methods

This chapter is concerned with the approximation of geodesic distances and geodesic curves with fast numerical schemes. This requires a discretization of the Riemannian manifold using either a uniform grid or a triangulated mesh. We focus on algorithms that compute the discrete distance by discretizing the Eikonal equation (1.24). The discrete non-linear problem can be solved by iterative schemes, and in some cases using faster front propagation methods.

We begin the description of these numerical algorithms by a simple setting in Section 2.3 where the geodesic distance is computed on a regular grid for an isotropic metric. This serves as a warmup for the general case studied in Section 2.4. This restricted setting is useful because the Eikonal equation is discretized using finite differences, which allows to introduce several important algorithms such as Gauss-Seidel iterations or Fast Marching propagations.

2.1 Eikonal Equation Discretization

This section describes a general setting for the computation of the geodesic distance. It follows the formulation proposed by Bornemann

and Rasch [33] that unifies several particular schemes, which are described in Sections 2.3 and 2.4.

2.1.1 Derivative-Free Eikonal Equation

The Eikonal equation (1.24) is a PDE that describes the infinitesimal behavior of the distance map U_S, it however fails to describe the behavior of U_S at points where it is not differentiable. To derive a numerical scheme for a discretized manifold one can consider an equation equivalent to (1.24) that does not involve derivatives of U_S.

We consider a small neighborhood $B(x)$ around each $x \in \Omega \backslash S$, such that $B(x) \cap S = \emptyset$. It can, for instance, be chosen as a Euclidean ball, see Figure 2.1.

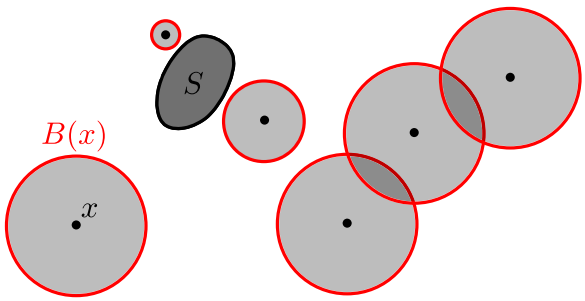


Fig. 2.1 *Neighborhood $B(x)$ for several points $x \in \Omega$.*

One can prove that the distance map U_S is the unique continuous solution U to the equation

$$\begin{cases} \forall\, x \in \Omega,\ U(x) = \min\limits_{y \in \partial B(x)} U(y) + d(y,x), \\ \forall\, x \in S,\ U(x) = 0, \end{cases} \tag{2.1}$$

where $d(y,x)$ is the geodesic distance defined in (1.15).

Equation (2.1) will be referred to as the control reformulation of the Eikonal equation. It makes appear that $U(x)$ depends only on values of U on neighbors y with $U(y) \leqslant U(x)$. This will be a key observation in order to speed up the computation of U.

The fact that U_S solves (2.1) is easy to prove. The triangular inequality implies that $U_S(x) \leqslant U_S(y) + d(y,x)$ for all $y \in \partial B(x)$. Con-

versely, the geodesic $\gamma^\star$ joining x to S passes through some $y \in \partial B(x)$, see Figure 2.2. This shows that this point y achieves the equality in the minimization (2.1).

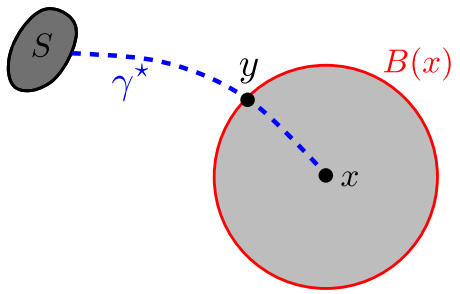


Fig. 2.2 *Proof of equation* (2.1).

Uniqueness is a bit more technical to prove. Let us consider two continuous mappings U_1 and U_2 that satisfy $U_1(x) \neq U_2(x)$ for some $x \notin S$. We define $\varepsilon = U_1(x) - U_2(x) \neq 0$ and $\gamma^\star$ a geodesic curve between S and x, such that $\gamma^\star(0) = x$ and $\gamma^\star(1) \in S$. We define $A = \{t \in [0,1] \backslash U_1(\gamma^\star(t)) - U_2(\gamma^\star(t)) = \varepsilon\}$. By definition we have $1 \notin A$ since $U_1(\gamma^\star(1)) = U_2(\gamma^\star(1)) = 0$. Furthermore, $0 \in A$, and this set is non-empty. Let us denote $s = \sup A$ its supremum. U_1 and U_2 being continuous mappings, we have $U_1(s) - U_2(s) = \varepsilon$ and $s \in A$, and thus $s \neq 1$ and $x_s = \gamma^\star(s) \notin S$. If we denote y an intersection of $\partial B(x_s)$ with the part of the geodesic $\gamma^\star$ joining x_s to S, we get $U_1(x_s) = U_1(y) + d(y, x_s)$ and $U_2(x_s) = U_2(y) + d(y, x_s)$, such that $U_1(y) - U_2(y) = \varepsilon$. Let us denote t such that $y = \gamma^\star(t)$. Then $t \in A$ and $t > s$, which contradicts the definition of s. Thus the initial hypothesis $U_1(x) \neq U_2(x)$ is false.

Equation (2.1) can be compactly written as a fixed point

$$U = \Gamma(U)$$

over the set of continuous functions U satisfying $U(x) = 0$ for all $x \in S$, where the operator $V = \Gamma(U)$ is defined as

$$V(x) = \min_{y \in \partial B(x)} U(y) + d(y, x).$$

2.1.2 Manifold Discretization

To compute numerically a discrete geodesic distance map, we suppose that the manifold Ω is sampled using a set of points $\{x_i\}_{i=0}^{N-1} \subset \Omega$. We denote ε the precision of the sampling.

The metric of the manifold is supposed to be sampled on this grid, and we denote

$$T_i = T_{x_i} \in \mathbb{R}^{2\times 2}$$

the discrete metric.

To derive a discrete counterpart to the Eikonal equation (1.24), each point x_i is connected to its neighboring points $x_j \in \text{Neigh}(x_i)$. Each point is associated with a small surrounding neighborhood $B_\varepsilon(x_i)$, that is supposed to be a disjoint union of simplexes whose extremal vertices are the grid points $\{x_i\}_i$. The sampling is assumed to be regular, so that the simplexes have approximately a diameter of ε.

For instance, in 2D, each neighborhood $B_\varepsilon(x_i)$ is an union of triangles

$$B_\varepsilon(x_i) = \bigcup_{\substack{x_j,x_k\in\text{Neigh}(x_i)\\ x_j\in\text{Neigh}(x_k)}} t_{i,j,k} \tag{2.2}$$

where $t_{i,j,k}$ is the convex hull of $\{x_i, x_j, x_k\}$.

Figure 2.3 shows example of 2D neighborhoods sets for two important situations. On a square grid, the points are equi-spaced, $x_i = (i_1\varepsilon, i_2\varepsilon)$, and each $B_\varepsilon(x_i)$ is composed of four regular triangles. On a triangular mesh, each $B_\varepsilon(x_i)$ is composed of the triangles which contain x_i.

This description extends to arbitrary dimensions. For instance, for a 3D manifold, each $B_\varepsilon(x_i)$ is an union of tetrahedra.

2.1.3 Discrete Eikonal Equation

A distance map $U_S(x)$ for $x \in \Omega$ is approximated numerically by computing a discrete vector $u \in \mathbb{R}^N$ where each value u_i is intended to approximate the value of $U_S(x_i)$.

This discrete vector is computed as a solution of a finite dimensional fixed point equation that discretizes (2.1). To that end, a continuous

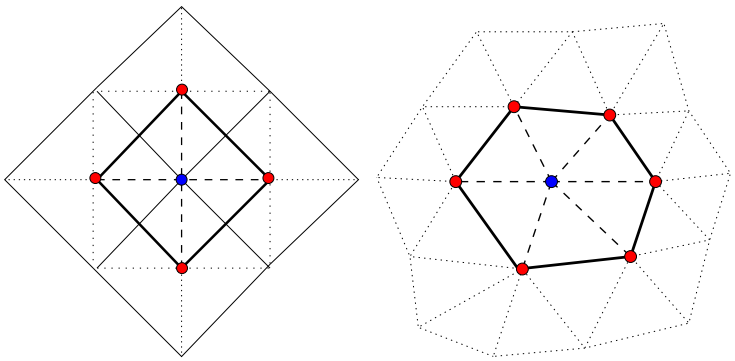


Fig. 2.3 *Neighborhood sets on a regular grid (left), and a triangular mesh (right)*

function $u(x)$ is obtained from the discrete samples $\{u_i\}_i$ by linear interpolation over the triangles.

We compute the minimization in (2.1) at the point $x = x_i$ over the boundary of $B_\varepsilon(x_i)$ defined in (2.2) where ε is the sampling precision. Furthermore, the tensor metric is approximated by a constant tensor field equal to T_i over $B_\varepsilon(x_i)$. Under these assumptions, the discrete derivative free Eikonal equation reads

$$\begin{cases} \forall\, x_i \in \Omega, & u_i = \min\limits_{y \in \partial B_\varepsilon(x_i)} u(y) + \|y - x_i\|_{T_i^{-1}}, \\ \forall\, x_i \in S, & u_i = 0. \end{cases} \tag{2.3}$$

Decomposing this minimization into each triangle (in 2D) of the neighborhood, and using the fact that $u(y)$ is affine on each triangle, one can re-write the discrete Eikonal equation as a fixed point

$$u = \Gamma(u) \quad \text{and} \quad \forall\, x_i \in S,\ u_i = 0 \tag{2.4}$$

where the operator $v = \Gamma(u) \in \mathbb{R}^N$ is defined as

$$v_i = \Gamma_i(u) = \min_{t_{i,j,k} \subset B_\varepsilon(x_i)} v_{i,j,k} \tag{2.5}$$

where

$$v_{i,j,k} = \min_{t \in [0,1]} t u_j + (1-t) u_k + \|t x_j + (1-t) x_k - x_i\|_{T_i^{-1}}.$$

We have written this equation for simplicity in the 2D case, so that each point y is a barycenter of two sampling points, but this extends

to a manifold of arbitrary dimension d by considering barycenters of d points.

Sections 2.3 and 2.4 show how this equation can be solved explicitly for the case of a regular square grid and for a 2D triangulation.

Convergence of the discretized equation. One can show that the fixed point equation (2.4) has a unique solution $u \in \mathbb{R}^N$, see [33]. Furthermore, if the metric T_x is continuous, one can also prove that the interpolated function $u(x)$ converges uniformly to the viscosity solution U_S of the Eikonal equation (1.24) when ε tends to zero. This was first proved by Rouy and Tourin [242] for the special case of an isotropic metric on a regular grid, see [33] for the general case.

2.2 Algorithms for the Resolution of the Eikonal Equation

The discrete Eikonal equation is a non-linear fixed point problem. One can compute the solution to this problem using iterative schemes. In some specific cases, one can compute the solution with non-iterative Fast Marching methods.

2.2.1 Iterative Algorithms

One can prove that the mapping Γ is both monotone

$$u \leqslant \tilde{u} \quad \Longrightarrow \quad \Gamma(u) \leqslant \Gamma(\tilde{u}), \tag{2.6}$$

and non-expanding

$$\|\Gamma(u) - \Gamma(\tilde{u})\|_\infty \leqslant \|u - \tilde{u}\|_\infty = \max_i |u_i - \tilde{u}_i|. \tag{2.7}$$

These two properties enable the use of simple iterations that converge to the solution u of the problem.

Jacobi iterations To compute the discrete solution of (2.4), one can apply the update operator Γ to the whole set of grid points. Jabobi non-linear iterations initialize $u^{(0)} = 0$ and then compute

$$u^{(k+1)} = \Gamma(u^{(k)}).$$

Properties (2.6) and (2.7), together with the fact that the iterates $u^{(k)}$ are bounded, imply the convergence of $u^{(k)}$ to a fixed point u satisfying (2.4). Algorithm 1 details the steps of the Jacobi iterations.

The fixed point property is useful to monitor the convergence of iterative algorithms, since one stops iterations when one has computed some distance u with

$$\|\Gamma(u) - u\|_\infty \leqslant \eta \quad \text{where} \quad \|u\|_\infty = \max_i |u_i|,$$

and where $\eta > 0$ is some user-defined tolerance.

Non-adaptive Gauss-Seidel iterations To speed up the computation, one can apply the local updates sequentially, which corresponds to a non-linear Gauss-Seidel algorithm, that converges to the solution of (2.4) – see Algorithm 2.

Adaptive Gauss-Seidel iterations. To further speed up the convergence of the Gauss-Seidel iterations, and to avoid unnecessary updates, Bornemann et al. introduced in [33] an adaptive algorithm that

Algorithm 1: Jacobi algorithm.

Initialization: set $u^{(0)} = 0$, $k \leftarrow 0$.
repeat
  for $0 \leqslant i < N$ **do**
    $u_i^{(k+1)} = \Gamma_i(u^{(k)})$.
  Set $k \to k+1$.
until $\|u^{(k)} - u^{(k-1)}\|_\infty < \eta$;

Algorithm 2: Non-adaptive Gauss-Seidel algorithm.

Initialization: set $u^{(0)} = 0$, $k \leftarrow 0$.
repeat
  Set $u^{(k+1)} = u^{(k)}$
  for $0 \leqslant i < N$ **do**
    $u_i^{(k+1)} = \Gamma_i(u^{(k+1)})$.
  Set $k \to k+1$.
until $\|u^{(k)} - u^{(k-1)}\|_\infty < \eta$;

maintains a list $\mathcal{Q}$ of points that need to be updated. At each iteration, a point is updated, and neighboring points are inserted to the list if they violate the fixed point condition up to a tolerance η. This algorithm is detailed in Algorithm 3.

See also [140, 287, 144] for other fast iterative schemes on parallel architectures.

2.2.2 Fast Marching Algorithm

The resolution of the discrete Eikonal equation (2.4) using the Gauss-Seidel method shown in Algorithm 2, is slow because all the grid points are visited several times until reaching an approximate solution.

For an isotropic metric on a regular grid, Sethian [254] and Tsitsiklis [276] discovered independently that one can by-pass these iterations by computing exactly the solution of (2.4) in $O(N \log(N))$ operations, where N is the number of sampling points. Under some conditions on the sampling grid and on the metric, this scheme extends to general discretized Riemannian manifolds, see Section 2.4.3.

This algorithm is based on an optimal ordering of the grid points that ensures that each point is visited only once by the algorithm, and that this visit computes the exact solution. For simplicity, we detail the algorithm for a 2D manifold, but it applies to arbitrary dimension.

Causality and ordering. A desirable property of the discrete update operator Γ is the following causality principle, that requires for

Algorithm 3: Adaptive Gauss Seidel algorithm.

Initialization: $u_i = \begin{cases} 0 \quad \text{if} \quad i \in S, \\ +\infty \text{ otherwise.} \end{cases} \quad \mathcal{Q} = S.$

repeat
  Pop i from $\mathcal{Q}$.
  Compute $v = \Gamma_i(u)$.
  if $|v - u_i| > \eta$ **then**
    Set $u_i \leftarrow v$.
    $\mathcal{Q} \leftarrow \mathcal{Q} \cup \text{Neigh}(x_i)$.
until $\mathcal{Q} = \emptyset$;

any v that

$$\forall\, x_j \in \text{Neigh}(x_i),\ \Gamma_i(v) \geqslant v_j. \tag{2.8}$$

This condition is a strong requirement that is not always fulfilled by a given manifold discretization, either because the sampling triangles have poor shapes, or because the metric T_i is highly anisotropic.

If this causality principle (2.8) holds, one can prove that the value u_i of the solution of the discrete Eikonal equation (2.4) at point x_i can be computed by using pairs of neighbors $x_j, x_k \in \text{Neigh}(x_i)$ with strictly smaller distance values

$$u_i > \max(u_j, u_k).$$

This property suggests that one can solve progressively for the solution u_i for values of u sorted in increasing order.

Front propagation. The Fast Marching algorithm uses a priority queue to order the grid points as being the current estimate of the distance. At a given step of the algorithm, each point x_i of the grid is labeled according to a state

$$\Sigma_i \in \{Computed, Front, Far\}.$$

During the iterations of the algorithm, while an approximation u_i of U_S is computed, a point can change of label according to

$$Far \mapsto Front \mapsto Computed.$$

Computed points with a state $\Sigma_i = Computed$ are those that the algorithm will not consider any more. This means that the computation of u_i is done for these points, so that $u_i \approx U_S(x_i)$. Front points x_i that satisfies $\Sigma_i = Front$ are the points being processed. The value of u_i is well defined but might change in future iterations. Far points with $\Sigma_i = Far$ are points that have not been processed yet, so that we define $u_i = +\infty$.

Because the value of $U_S(x_i)$ depends only on the neighbors x_i which have a smaller value, and because each update is guaranteed to only decrease the value of the estimated distance, the point in the front with the smallest current distance u_i is actually the point with smaller

distance U_S amongst the points in $Front \cup Far$. Selecting at each step this point thus guarantees that u_i is the correct value of the distance, and that its correct priority has been used.

Algorithm 4 gives the details of the front propagation algorithm that computes a distance map u approximating $U_S(x)$ on a discrete grid.

Numerical complexity. The worse case numerical complexity of this algorithm is $O(N \log(N))$ for a discrete grid of N points. This is because all the N points are visited (tagged *Computed*) exactly once while the time required for updating only depends on the size of the neighborhood. Furthermore, the selection of point i with minimum u_i from the priority queue of the front points takes at most $\log(N)$ operations with a special heap data structure [288, 119]. However, in practice, it takes much less time and the algorithm is nearly linear in time.

Using different data structures requiring additional storage, different $O(N)$ implementations of the Fast Marching were proposed in [144, 293].

Algorithm 4: Fast Marching algorithm.

Initialization:
$\forall\, x_i \in S, u_i \leftarrow 0,\ \Sigma_i \leftarrow Front,$
$\forall\, x_i \notin S, u_i \leftarrow +\infty,\ \Sigma_i \leftarrow Far.$
repeat
  Select point: $i \longleftarrow \underset{k, \Sigma_k = Front}{\operatorname{argmin}} u_k.$
  Tag: $\Sigma_i \leftarrow Computed.$
  for $x_k \in \mathrm{Neigh}(x_i)$ **do**
    if $\Sigma_k = Far$ **then**
      $\Sigma_k \leftarrow Front$
    if $\Sigma_k = Front$ **then**
      $u_k \leftarrow \Gamma_k(u).$
until $\{i \setminus \Sigma_i = Front\} = \emptyset$;

2.2.3 Geodesics on Graphs

Graph as discrete manifold. In this setting, for each $x_j \in \text{Neigh}(x_i)$, the metric is represented as a weight $W_{i,j}$ along the edge $[x_i, x_j]$. The graph is an abstract structure represented only by its indices $0 \leqslant i < N$, and by its weights. We denote $i \sim j$ to indicate that the points are connected for $x_j \in \text{Neigh}(x_i)$. One only manipulates the indices i of the points x_i for the geodesic computation on the graph with the position x_i being used only for display purpose.

One should be careful that in this graph setting, the metric $W_{i,j}$ is defined on the edge $[x_i, x_j]$ of the graph, whereas for the Eikonal equation discretization detailed in Section 2.1.2, the metric is discretized on the vertex x_i. Notice that – while it is less usual – it is possible to define graphs with weights on the vertices rather than on the edges.

Geodesic distances on graphs. A path γ on a graph is a set of K_γ indices $\{\gamma_t\}_{t=0}^{K_\gamma-1} \subset \Omega$, where $K_\gamma > 1$ can be arbitrary, $0 \leqslant \gamma_t < N$, and each edge is connected on the graph, $\gamma_t \sim \gamma_{t+1}$. The length of this path is

$$L(\gamma) = \sum_{t=0}^{K_\gamma-2} W_{\gamma_t,\gamma_{t+1}}.$$

The set of path joining two indices is

$$\mathcal{P}(i,j) = \left\{\gamma \setminus \gamma_0 = i \quad \text{and} \quad \gamma_{K_\gamma-1} = j\right\},$$

and the geodesic distance is

$$d_{i,j} = \min_{\gamma \in \mathcal{P}(i,j)} L(\gamma).$$

Floyd Algorithm. A simple algorithm to compute the geodesic distance $d_{i,j}$ between all pairs of vertices is the algorithm of Floyd [249]. Starting from an initial distance map

$$d_{i,j}^{(0)} = \begin{cases} W_{i,j} & \text{if} \quad x_j \in \text{Neigh}(x_i), \\ +\infty & \text{otherwise,} \end{cases}$$

it iterates, for $k = 0, \ldots, N-1$

$$d_{i,j}^{(k+1)} = \min_l (d_{i,j}^{(k)}, d_{i,l}^{(k)} + d_{l,j}^{(k)}).$$

One can then show that $d_{i,j}^{(N)} = d_{i,j}$. The complexity of this algorithm is $O(N^3)$ operations, whatever the connectivity of the graph is.

Dijkstra algorithm. The Fast Marching algorithm can be seen as a generalization of Dijkstra algorithm that computes the geodesic distance on a graph [101]. This algorithm computes the distance map to an initial vertex i_s

$$u_j = d_{i_s,j}$$

using a front propagation on the graph. It follows the same steps of the Fast Marching as detailed in Algorithm 4.

The update operator (2.5) is replaced by an optimization along the adjacent edges of a vertex

$$\Gamma_i(u) = \min_{j \sim i} u_j + W_{i,j} \tag{2.9}$$

As for the Fast Marching algorithm, the complexity of this algorithm is $O(VN + N\log(N))$, where V is a bound on the size of the one ring $\text{Neigh}(x_i)$. For sparse graphs, where V is a small constant, computing the distance between all pairs of points in the graph thus requires $O(N^2 \log(N))$ operations, which is significantly faster than Floyd algorithm.

Geodesics on graph. Once the distance map u to some starting point i_s has been computed using Dijkstra algorithm, one can compute the shortest path γ between i_s and any points i_e by performing a discrete gradient descent on the graph

$$\gamma_0 = i_e \quad \text{and} \quad \gamma_{t+1} = \operatorname*{argmin}_{i \sim \gamma_t} u_i.$$

Metrication error. As pointed out in [70] and [214], the distance u_i computed with this Dijkstra algorithm is however not a faithful discretization of the geodesic distance, and it does not converge to U_S when N tends to $+\infty$. For instance, for a Euclidean metric $W(x) = 1$, the distance between the two corners $x_s = (0,0)$ and $x_e = (1,1)$ computed with Dijkstra algorithm is always the Manhattan distance $d(x_s, x_e) = \|x_s - x_e\|_1 = 2$ whereas the geodesic distance should be

$\|x_s - x_e\| = \sqrt{2}$. This corresponds to a metrication error, which can be improved but not completely removed by an increase of the size of the chosen neighborhood.

One can however prove that randomized refinement rule produces a geodesic distance on graph that converges to the geodesic distance on the manifold, see for instance [39, 26]. This requires that the vertices of the graph are a dense covering of the manifold as detailed in Section 4.2.1. This also requires that the edges of the graph links all pairs of vertices that are close enough.

2.3 Isotropic Geodesic Computation on Regular Grids

This section details how the general scheme detailed in Section 2.1 is implemented in a simple setting relevant for image processing and volume processing.

The manifold is supposed to be sampled on a discrete uniform grid, and the metric is isotropic. We consider here only the 2D case to ease notations so that the sampling points are $x_i = x_{i_1,i_2} = (i_1\varepsilon, i_2\varepsilon)$, and the metric reads $T_i = W_i \mathrm{Id}_2$. The sampling precision is $\varepsilon = 1/\sqrt{N}$, where N is the number of pixels in the image.

2.3.1 Update Operator on a Regular Grid

Each grid point x_i is connected to four neighbors $\mathrm{Neigh}(x_i)$, see Figure 2.3, left, and the discrete update operator $\Gamma_i(u)$ defined in (2.5) computes a minimization over four adjacent triangles.

$$\Gamma_i(u) = \min_{t_{i,j,k} \subset B_\varepsilon(x_i)} v_{i,j,k} \tag{2.10}$$

where

$$v_{i,j,k} = \min_{t \in [0,1]} tu_j + (1-t)u_k + W_i \|t x_j + (1-t)x_k - x_i\|. \tag{2.11}$$

This corresponds to the minimization of a strictly convex function under convex constraints, so that there is a single solution.

As illustrated in figure 2.4, we model u as an affine mapping over

triangle $t_{i,j,k}$, and we denote

$$\begin{cases} t^\star = \text{argmin}_{t\in[0,1]}\{tu_j + (1-t)u_k + W_i\|tx_j + (1-t)x_k - x_i\|\} \\ x^\star = t^\star x_j + (1-t^\star)x_k \\ v^\star = t^\star u_j + (1-t^\star)u_k = u(x^\star) \end{cases}, \tag{2.12}$$

such that

$$\frac{v_{i,j,k} - v^\star}{\|x_i - x^\star\|} = W_i. \tag{2.13}$$

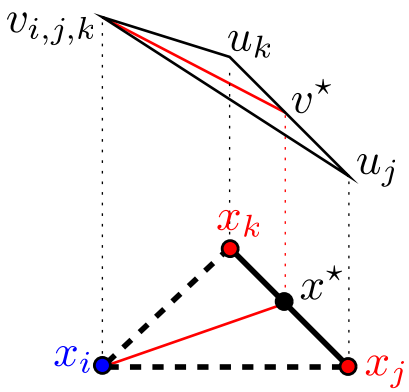


Fig. 2.4 *Geometric interpretation of update on a single triangle.*

From a geometrical point of view, finding $v_{i,j,k}$ and x^* is related to finding the maximal slope in $t_{i,j,k}$. If $\frac{|u_j-u_k|}{\|x_j-x_k\|} \leqslant W_i$, it is possible to find $x^* \in [x_j, x_k]$ such that equation 2.13 holds, and W_i is then the maximal slope in $t_{i,j,k}$. In this case, we have $\|\nabla u\| = W_i$, which can be rewritten as

$$(v_{i,j,k} - u_j)^2 + (v_{i,j,k} - u_k)^2 = \varepsilon^2 W_i^2. \tag{2.14}$$

Notice that the condition $\frac{|u_j-u_k|}{\|x_j-x_k\|} \leqslant W_i$ imposes that this equation has solutions. Furthermore, since equation (2.11) imposes that $v_{i,j,k}$ be larger than both u_j and u_k, $v_{i,j,k}$ corresponds to the largest root of (2.14).

If however $\frac{|u_j-u_k|}{\|x_j-x_k\|} > W_i$, including the cases when $u_j = +\infty$ or $u_k = +\infty$, W_i is no longer the maximal slope in $t_{i,j,k}$, and the solution of (2.11) is reached for $t = 0$ if $u_k < u_j$, and $t = 1$ if $u_j < u_k$.

Finally, $v_{i,j,k}$ is computed as

$$v_{i,j,k} = \begin{cases} \frac{1}{2}(u_j + u_k + \sqrt{\Delta}) & \text{if} \quad \Delta \geqslant 0, \\ \min(u_j, u_k) + \varepsilon W_i & \text{otherwise} \end{cases} \tag{2.15}$$

where

$$\Delta = 2\varepsilon^2 W_i^2 - (u_j - u_k)^2$$

Optimizing computation. The equivalence of the four triangles neighboring x_i suggests that some computations can be saved with respect to the expression (2.10) of the update operator. As an example, if $u_{i_1-1,i_2} > u_{i_1+1,i_2}$, the solution computed from triangle $t_{i,(i_1-1,i_2),(i_1,i_2+1)}$ will be larger than the one computed from triangle $t_{i,(i_1+1,i_2),(i_1,i_2+1)}$, i.e. $v_{i,(i_1-1,i_2),(i_1,i_2+1)} > v_{i,(i_1+1,i_2),(i_1,i_2+1)}$. Computing $v_{i,(i_1-1,i_2),(i_1,i_2+1)}$ is thus unnecessary.

More generally, denoting, for a given point x_i

$$v_1 = \min(u_{i_1-1,i_2}, u_{i_1+1,i_2}) \quad \text{and} \quad v_2 = \min(u_{i_1,i_2-1}, u_{i_1,i_2+1}),$$

the update operator $v = \Gamma_i(u)$ is obtained as

$$\Gamma_i(u) = \begin{cases} \frac{1}{2}(v_1 + v_2 + \sqrt{\Delta}) & \text{if} \quad \Delta \geqslant 0, \\ \min(v_1, v_2) + \varepsilon W_i & \text{otherwise.} \end{cases}$$

where Δ corresponds to solving the equation (2.11) in the triangle with minimal values. This update scheme can be found in [70, 77] for 2D Fast Marching and was extended to 3D images in [93].

2.3.2 Fast Marching on Regular Grids

One can prove that the update operator defined by (2.10) satisfies the causality condition (2.8). One can thus use the Fast Marching algorithm, detailed in Algorithm 4, to solve the Eikonal equation in $O(N \log(N))$ operations, where N is the number of grid points.

Figure 2.5 shows examples of Fast Marching propagations on a regular 2D grid.

Other methods, inspired by the Fast Marching methods, have been developed, such as the Fast Sweeping [275]. They can be faster in some cases, and implemented on parallel architectures [298].

Alternative discretization schemes, which might be more precise in some cases, have been proposed, that can also be solved with Fast Marching methods [228, 236, 64, 202].

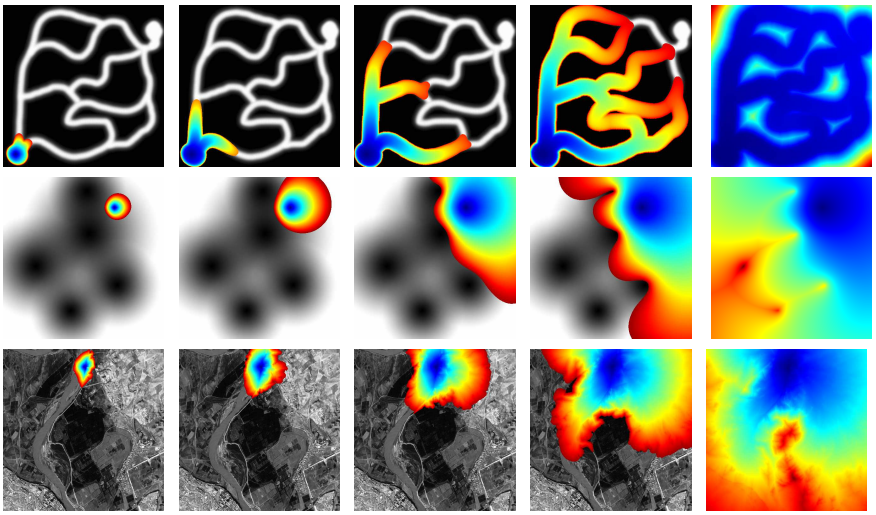


Fig. 2.5 *Examples of isotropic front propagation. The color-map indicates the values of the distance functions at a given iteration of the algorithm. The background image is the potential W, which range from 10^{-2} (white) to 0 (black), so that geodesics tend to follow bright regions.*

Reducing Computational Time. Notice also that the computational time of algorithm 4 can be reduced in the following way : when x_j and $x_k \in \text{Neigh}(x_i)$ are selected and $\Sigma_k = Front$, computing $\Gamma_k(u)$ is not always needed in order to update u_k. Indeed, if we denote x_j the symetric of x_i with respect to x_k and if $u_j < u_i$, then $v_1 = u_j$ or $v_2 = u_j$ during the computation of $\Gamma_k(u)$, and u_i will not be used. In this case, it is thus unnecessary to update u_k. Overall, roughly half of the computations can be saved in that way.

Fast Marching Inside Shapes. It is possible to restrict the propagation inside an arbitrary compact sub-domain Ω of $\mathbb{R}^d$. This is achieved numerically by removing a connection $x_j \in \text{Neigh}(x_i)$ if the segment $[x_i, x_j]$ intersect the boundary $\partial\Omega$.

Figure 2.6 shows an example of propagation inside a planar domain $\Omega \subset \mathbb{R}^2$.

Chapter 5 details applications of geodesic computations inside a planar domain to perform shape comparison and retrieval.

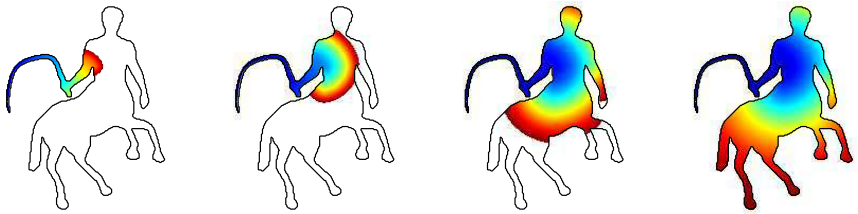


Fig. 2.6 *Fast Marching propagation inside a 2D shape.*

2.3.3 Upwind Finite Differences

As suggested by equation (2.14), this update step can be reformulated in the more classical framework of upwind finite differences, which is a usual tool to discretize PDE. One needs to be careful about the discretization of the gradient operator, such that the minimal solution over all the triangles is selected.

Upwind Discretization. For a discrete 2D image u_i sampled at location $x_i = (i_1\varepsilon, i_2\varepsilon)$, we denote $u_i = u_{i_1,i_2}$ to ease the notations. Forward or backward finite differences are defined as

$$\begin{array}{llll} (D_1^+u)_i = (u_{i_1+1,i_2} - u_{i_1,i_2})/\varepsilon & \text{and} & (D_1^-u)_i = (u_{i_1,i_2} - u_{i_1-1,i_2})/\varepsilon \\ (D_2^+u)_i = (u_{i_1,i_2+1} - u_{i_1,i_2})/\varepsilon & \text{and} & (D_2^-u)_i = (u_{i_1,i_2} - u_{i_1,i_2-1})/\varepsilon. \end{array}$$

Upwind schemes, initially proposed by Rouy and Tourin [242] for the Eikonal equation, retain the finite difference with the largest magnitude. This defines upwind partial finite differences

$$\begin{aligned} (\tilde{D}_1u)_i &= \max((D_1^+u)_i, -(D_1^-u)_i, 0), \quad \text{and} \\ (\tilde{D}_2u)_i &= \max((D_2^+u)_i, -(D_2^-u)_i, 0), \end{aligned}$$

where one should be careful about the minus sign in front of the backward derivatives. This defines an upwind gradient

$$(\tilde{\nabla} u)_i = ((\tilde{D}_1 u)_i, (\tilde{D}_2 u)_i) \in \mathbb{R}^2.$$

The discrete Eikonal equation can be written as

$$\|(\tilde{\nabla} u)_i\| = W_i \quad \text{with} \quad \forall\, x_i \in S,\ u_i = 0. \tag{2.16}$$

In the restricted isotropic setting, this equation is the same as (2.5).

2.4 Anisotropic Geodesic Computation on Triangulated Surfaces

Uniform grids considered in the previous section are too constrained for many applications in vision and graphics. To deal with complicated domains, this section considers planar triangulations in $\mathbb{R}^2$ and triangulated surfaces in general dimension (e.g. $\mathbb{R}^3$), which are two important settings of 2D Riemannian manifolds of dimension 2. In particular, we consider generic metrics T_x that are not restricted to be isotropic as in the previous section. The anisotropy of the metric raises convergence issues.

The techniques developed in this section generalize without difficulties to higher dimensional manifolds by considering higher order simplexes instead of triangles. For instance tetrahedra should be used to discretize 3D manifolds.

2.4.1 Triangular Grids

We consider a triangulation

$$\Omega = \bigcup_{(i,j,k)\in\mathcal{T}} t_{i,j,k}$$

of the manifold, where each sampling point $x_i \in \mathbb{R}^d$, and each triangle $t_{i,j,k}$ is the convex hull of (x_i, x_j, x_k). The set of triangle indices is $\mathcal{T} \subset \{0, \ldots, N-1\}^3$. Each edge (x_i, x_k) of a face belongs either to two different triangles, or to a single triangle for edges on the boundary of the domain.

We consider for each vertex x_i a tensor $T_i \in \mathbb{R}^{d\times d}$. This section considers both planar domains Ω, and domains that are surfaces $\Omega \subset \mathbb{R}^d$ equipped with a metric T_i that operates in the tangent plane. A careful design of the tensor field T_i makes the treatment of these two settings amendable to the same algorithms in a transparent manner.

Planar manifolds. For a planar triangulation, $d = 2$, the set of triangles $\{t_{i,j,k}\}_{(i,j,k)\in\mathcal{T}}$ is a partition of the domain $\Omega \subset \mathbb{R}^2$, possibly with an approximation of the boundary. Each vertex x_i is associated with a 2D planar tensor $T_i \in \mathbb{R}^{2\times 2}$ that discretizes the Riemannian metric.

We note that this planar triangulation framework also allows one to deal with anisotropic metrics on an image (a regular grid), by splitting each square into two adjacent triangles.

Surface manifolds. We also consider the more general setting of a discrete surface $\mathcal{S}$ embedded in $\mathbb{R}^3$, in which case $\mathcal{S}$ is a discretization of a continuous surface in $\mathbb{R}^3$. In this case, the tensor $T_i \in \mathbb{R}^{3\times 3}$ is intended to compute the length $\|\gamma'\|_{T_i}$ of the tangent $\gamma'(t)$ to a curve γ traced on $\mathcal{S}$. These tangents are vector $\gamma'(t) \in \mathcal{T}_{x_i}$, the 2D tangent plane at $x_i = \gamma(t)$ to $\mathcal{S}$. We thus assume that the tensor T_i is vanishing for vectors v orthogonal to the tangent plane $\mathcal{T}_{x_i}$, $T_i v = 0$. This corresponds to imposing that the tensor is written as

$$T_i = \lambda_1(x_i) e_1(x_i) e_1(x_i)^{\mathrm{T}} + \lambda_2(x_i) e_2(x_i) e_2(x_i)^{\mathrm{T}}, \tag{2.17}$$

where $(e_1(x_i), e_2(x_i))$ is an orthogonal basis of $\mathcal{T}_{x_i}$.

2.4.2 Update Operator on Triangulated Surfaces

One can compute explicitly the update operator (2.4) on a 2D triangulation. This update rule was initially proposed by Kimmel and Sethian [152] and Fomel [120] for surface meshes with isotropic metric, and was extended to anisotropic metrics in [40]. The process is essentially the same as in Section 2.3.1. The formulation we propose has the advantage of being easily generally applicable to higher dimension manifolds.

The discrete update operator $\Gamma_i(u)$ defined in (2.5) computes a minimization over all adjacent triangles.

$$\Gamma_i(u) = \min_{t_{i,j,k} \subset B_\varepsilon(x_i)} v_{i,j,k}$$

where

$$v_{i,j,k} = \min_{t \in [0,1]} t u_j + (1-t) u_k + \| t x_j + (1-t) x_k - x_i \|_{T_i^{-1}}. \tag{2.18}$$

Let us denote

$$p = (u_j, u_k)^{\mathrm{T}} \in \mathbb{R}^2,\ \mathbb{I} = (1,1)^{\mathrm{T}} \in \mathbb{R}^2 \text{ and } X = (x_i - x_j, x_i - x_k) \in \mathbb{R}^{2\times 2}.$$

Modeling u as an affine mapping over triangle $t_{i,j,k}$, and defining $X^+ = X(X^{\mathrm{T}}X)^{-1}$, one can show as in Section 2.3.1 that under the condition

$$\Delta = \|X^+\mathbb{I}\|^2_{T_i^{-1}} + \langle X^+\mathbb{I},\, X^+p\rangle_{T_i^{-1}} - \|X^+\mathbb{I}\|^2_{T_i^{-1}} \|X^+p\|^2_{T_i^{-1}} \geqslant 0,$$

the solution v of (2.18) is the largest root of the following quadratic polynomial equation,

$$v^2 \|X^+\mathbb{I}\|^2_{T_i^{-1}} - 2v\langle X^+\mathbb{I},\, X^+p\rangle_{T_i^{-1}} + \|X^+p\|_{T_i^{-1}} = 1.$$

If $\Delta < 0$, the update is achieved by propagating from x_j or x_k. The update operator for the triangle $t_{i,j,k}$ is thus defined as

$$v_{i,j,k} = \begin{cases} \dfrac{\langle X^+\mathbb{I}, X^+p\rangle_{T_i^{-1}} + \sqrt{\Delta}}{\|X^+\mathbb{I}\|^2_{T_i^{-1}}} & \text{if} \quad \Delta \geqslant 0, \\ \min(u_j + \|x_j - x_i\|_{T_i^{-1}}, u_k + \|x_k - x_i\|_{T_i^{-1}}) & \text{otherwise.} \end{cases} \tag{2.19}$$

Note that these calculations are developed in [214], which also genelarizes them to an arbitrary dimension.

2.4.3 Fast Marching on Triangulation

Unfortunately, the update operator (2.10) on a triangulated mesh does not satisfy in general the causality requirement (2.8).

One notable case in which this condition holds is for an isotropic metric and a triangulated surface in $\mathbb{R}^3$ that does not contain poorly shaped triangles with obtuse angles. In this case, one can use the Fast

Marching algorithm, detailed in Algorithm 4 to solve the Eikonal equation in $O(N \log(N))$ operations, where N is the number of grid points. Figure 2.7 shows an example of propagation on a triangulated surface, for a constant isotropic metric. The colored region corresponds to the points that have been computed at a step of the propagation with its boundary being the front.

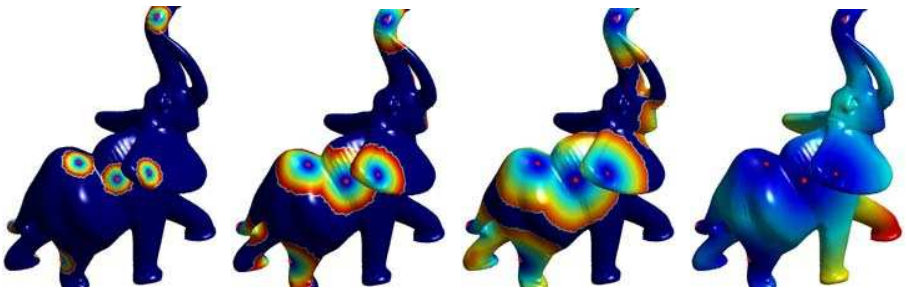


Fig. 2.7 *Example of Fast Marching propagation on a triangulated mesh.*

Reducing Computational Time. As in the case of regular grids, computational time of algorithm 4 can be reduced : let us assume that x_i and $x_k \in \text{Neigh}(x_i)$ were selected, and that $\Sigma_k = Front$. During the computation of $\Gamma_k(u)$, $v_{k,j,m}$ needs not be computed if x_i is not a vertice of $t_{k,j,m}$. Indeed, the value of $v_{k,j,m}$ is either $+\infty$, or has not changed since it was last computed. Omitting such calculations can lead to an important computational gain, depending on the connectivity of the mesh.

Triangulations with Strong Anisotropy or Obtuse Angles. If the triangulation is planar with strong anisotropy in the metric, or if the triangulation contains obtuse angles (these two conditions being essentially dual, as shown in [214]), then the Fast Marching method might fail to compute the solution of the discrete Eikonal equation.

In this case, several methods have been proposed to obtain such a solution. Firstly, one can split the obtuse angle by extending the neighborhood of a point if it contains obtuse angles [153], see Figure 2.8.

However, computing the neighbor n to add is a non-trivial task, even in dimension 2, and extending the neighborhood has several drawbacks : loss of precision, loss of the topology of the manifold, increase of the running-time (depending on the anisotropy of the tensor, or on the measure of the obtuse angle to split).

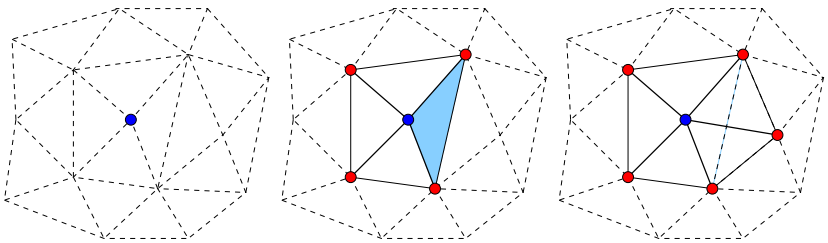


Fig. 2.8 *Obtuse angle splitting : assume that the blue point x_i on the left image is to be updated. As shown on the middle image, its natural neighborhood $B_\varepsilon(x_i)$ with respect to the triangulation which represents the surface contains an obtuse angle. In order to perform the update, a farther point y needs to be added to $B_\varepsilon(x_i)$ such that $B_\varepsilon(x_i) \cup \{y\}$ does not contain an obtuse angle anymore.*

In the specific case when the manifold is completely parametrized from $\Omega = [0,1]^2$ or $\Omega = [0,1]^3$, the computation of neighbor n can be performed faster using integer programming [263, 46].

Early proposals to compute geodesic distances on surfaces make use of a level set implementation of the front propagation [148, 150].

The idea of extending the size of $B_\varepsilon(x_i)$ is more systematically developed in *Ordered Upwind Methods* (OUM) [256]. In OUM the 1-pixel width front of the Fast Marching is replaced by an adaptive width front, which depends on the local anisotropy. Notice that OUM is in fact a class of numerical methods which allows to solve a large class of partial differential equations. More specifically, its convergence is proven for all static Hamilton-Jacobi equations.

Another approach consists in running the standard Fast Marching, but to authorize a recursive correction of *Computed* points [155]. However, there is no proof of convergence for this approach, and the amount of calculations to perform the correction again depends on the anisotropy.

The fast sweeping method also extends to solve Eikonal equations

on triangulations [233] and works under the same condition as the Fast Marching.

2.5 Computing Minimal Paths

2.5.1 Geodesic Curves Extraction

Once the discrete geodesic distance u approximating U_S on the computation grid is computed, the geodesic $\gamma^\star$ between some point x_e and S is extracted by integrating numerically the ordinary differential equation (ODE) (1.26).

Precision of the numerical computation of geodesics. If x_e is not in MedAxis(S), one can prove that the continuous geodesic $\gamma^\star(t)$ never crosses MedAxis(S) for $t \in [0,1]$, so that the distance map U_S is smooth along $\gamma^\star(t)$ and (1.26) makes sense.

The precision of the discrete geodesic, and its deviation from the continuous curve depends on the distance of x_e to MedAxis(S). As x_e approaches MedAxis(x_s), small approximation errors on U_S can lead to important errors in the approximation of $\gamma^\star$. Figure 2.9 shows how a small shift of the location of x_e leads to a completely different geodesic curve.

Thresholding distance maps. The geodesic $\gamma^\star$ between two points x_s and x_e satisfies

$$\{\gamma^\star(t) \setminus t \in [0,1]\} = \{x \in \Omega \setminus d(x_s,x) + d(x_e,x) = d(x_s,x_e)\}\,.$$

As was proposed in [147, 70], one can compute numerically the two distance maps U_{x_s} and U_{x_e}, and approximate the geodesic curve as

$$\{x \in \Omega \setminus |U_{x_s}(x) + U_{x_e}(x) - U_{x_e}(x_s)| \leqslant \varepsilon\} \tag{2.20}$$

where $\varepsilon > 0$ should be adapted to the grid precision and to the numerical errors in the computation of the distance maps. The thresholding (2.20) defines a thin band that contains the geodesic. Note that this method extends to compute the shortest geodesic between two sets S_1 and S_2.

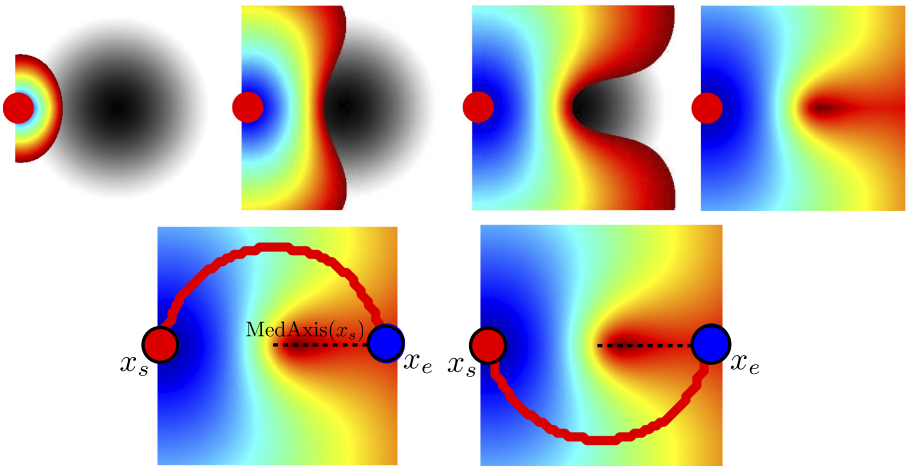


Fig. 2.9 *Top row: fast marching propagation in* $[0,1]^2$ *with a metric large in the center of the image. Bottom row: computation of shortest paths.*

Piecewise paths on triangulation. As detailed in Section 2.4, the geodesic distance map U_S is approximated on triangulations using a piecewise affine function. The gradient ∇U_S is thus constant inside each triangle.

A faithful numerical scheme to integrate numerically the ODE (1.26) computes a piecewise linear path $\gamma^\star$ that is linear inside each triangle. The path either follows an edge between two triangles, or is parallel to ∇U_S inside a triangle.

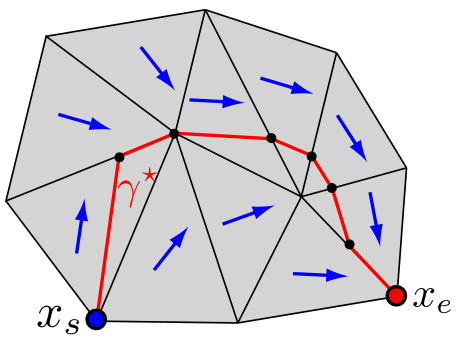


Fig. 2.10 *Piecewise linear geodesic path on a triangulation.*

Figure 2.10 shows an example of discrete geodesic path on a triangulation.

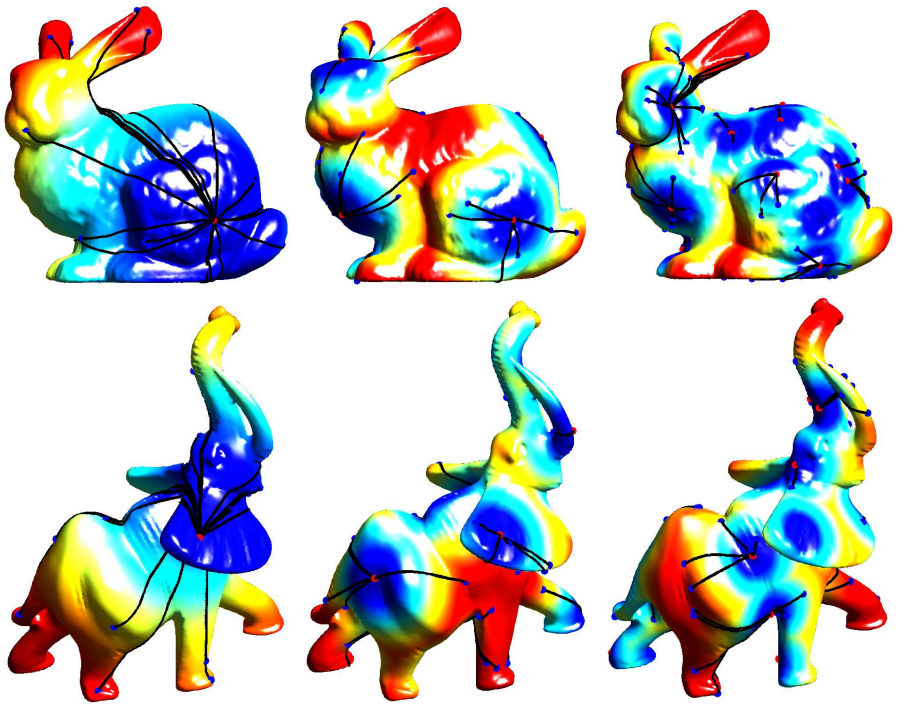


Fig. 2.11 *Examples of geodesic extraction on a mesh with, from left to right, an increasing number of starting points.*

Figure 2.11 shows examples of minimal paths extracted on a triangulated surface. It makes use of an increasing number of starting points S, so that a geodesic curve $\gamma^\star$ links a point $x_s \in S$ to its closest point in S.

Higher order ODE integration schemes. To produce smooth paths, one can use classical ODE integration schemes of arbitrary order, such as Runge-Kutta [113], to integrate the ODE (1.26). This approach is mostly used on regular grid, because ∇U_S can be computed using finite differences of arbitrary precision, and one can then use spline interpolation to obtain a smooth interpolated gradient field suitable for

arbitrary ODE integrators.

On complicated domains Ω, this requires a proper interpolation scheme near the boundary to ensure that the gradient always points inside the domain, and the discrete geodesic is well defined and stays inside Ω.

In practice, it is difficult to ensure this condition. A simple heuristic to construct a valid interpolated gradient field is to compute the geodesic distance map U_S on a small band outside the domain. The width of the extension of the domain should match the order of the interpolation scheme. This ensures that the finite differences are performed over valid stencils even for points along the boundary of the domain.

2.5.2 Joint Propagations

To extract a geodesic $\gamma^\star$ between two points x_s and x_e in Ω, it is sufficient to compute the geodesic distance U_{x_s} to x_s until the front propagation reaches x_e. Indeed since a gradient descent is then used to track the geodesic, the geodesic distance is decreasing from x_e to x_s, and we need only to know U_{x_s} where it is lower than $U_{x_s}(x_e)$.

It is possible to speed up the Fast Marching computation by starting simultaneously the front propagation from the two points, in order to compute the geodesic distance U_S from $S = \{x_s, x_e\}$. This method was first proposed for graphs in [223] using the Dijkstra propagation detailed in Section 2.2.3. It was used for finding a minimal path between two points with Fast Marching in [93]. The path was found as the union of its two halves through the use of the meeting point, which is a saddle point as shown in [70].

Saddle point. One can perform the Fast Marching algorithm until the fronts emanating from x_s and x_e meet. If $\gamma^\star$ is unique, which is the case except in degenerated situations, the fronts will meet at the saddle point $x_{s,e}$, which is the intersection of the geodesic mediatrix and $\gamma^\star$

$$x_{s,e} = \gamma^\star(t) \quad \text{where} \quad U_{x_s}(\gamma^\star(t)) = U_{x_e}(\gamma^\star(t)). \tag{2.21}$$

Saddle point $x_{s,e}$ is the point of minimal distance to both x_s and x_e on their mediatrix. The geodesic $\gamma^\star$ is the union of the geodesics

$$\gamma_s^\star \in \mathcal{P}(x_s, x_{s,e}) \quad \text{and} \quad \gamma_e^\star \in \mathcal{P}(x_e, x_{s,e})$$

between the saddle point and the two boundary points.

Joint propagation. One can thus stop the Fast Marching propagation to compute U_S when it reaches $x_{s,e}$. The front has then covered the union of two geodesic balls

$$\{x \in \Omega \setminus U_S(x) \leqslant U_S(x_{s,e})\} = R_s \cup R_e \tag{2.22}$$

where

$$\forall\, i \in \{s, e\}, \quad R_i = \{x \in \Omega \setminus U_{x_i}(x) \leqslant r\}$$

and $r = U_{x_s}(x_{s,e}) = U_{x_e}(x_{s,e})$. This is an important saving with respect to computing U_{x_s} with a propagation starting from x_s, which covers the larger region

$$\{x \in \Omega \setminus U_{x_s}(x) \leqslant U_{x_s}(x_e)\} = R.$$

For instance, as was proposed in [93], if the metric is Euclidean $T_x = \mathrm{Id}_d$ in $\mathbb{R}^d$, then R_s, R_e and R are spheres, and the computation gain is

$$\frac{|R|}{|R_s \cup R_e|} = 2^{d-1}.$$

Half geodesics. To extract the geodesics $\gamma_i^\star$ for $i \in \{s, e\}$, one needs to perform a gradient descent of U_S starting from $x_{s,e}$. Unfortunately $x_{s,e}$ is on the medial axis of S, which corresponds to the geodesic mediatrix. The distance map U_S is thus not differentiable at $x_{s,e}$. It is however differentiable on both side of the mediatrix, since it is equal to U_{x_i} in each region R_i for $i \in \{s, e\}$, and one can thus use as gradient $\nabla U_{x_i}(x_{s,e})$ to find each half of the geodesic. The two minimal paths are then obtained by the following gradient descent,

$$\begin{cases} \forall\, t > 0,\ \dfrac{\mathrm{d}\gamma_i^\star(t)}{\mathrm{d}t} = -\eta_t \nabla U_S(\gamma_i^\star(t)), \\ \dfrac{\mathrm{d}\gamma_i^\star(0)}{\mathrm{d}t} = -\eta_0 \nabla U_{x_i}(x_{s,e}) \quad \text{and} \quad \gamma^\star(0) = x_{s,e}. \end{cases}$$

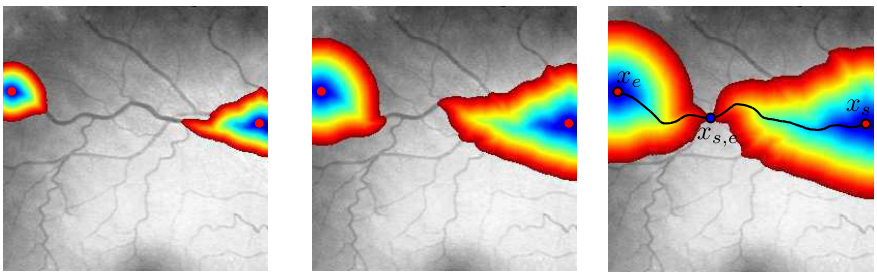


Fig. 2.12 *Example of joint propagations to extract a geodesic between two points $x_s, x_e \in \Omega$. The red points are the boundary points x_s, x_e, the blue point is the saddle point $x_{s,e}$.*

where the gradient step size η_t can be defined as in (1.26).

Figure 2.12 shows an example of joint propagation on an isotropic metric $T_x = W(x)^2 \mathrm{Id}_2$ to extract a geodesic that follows vessels in a medical image.

2.5.3 Heuristically driven propagations

To compute the geodesic between two points x_s, x_e, the Fast Marching algorithm explores a large region. Even if one uses the joint propagation, the front is located in two geodesic balls $R_s \cup R_e$ defined in (2.22). A large amount of computation is spent in regions that are thus quite far from the actual geodesic $\gamma^\star \in \mathcal{P}(x_s, x_e)$.

It is possible to use heuristics to drive the propagation and restrict the front to be as close as possible from the minimal path $\gamma^\star$ one wishes to extract.

Heuristics and propagation. We consider the Fast Marching algorithm to compute the distance $u \in \mathbb{R}^N$, which is intended to be an approximation of the continuous distance U_{x_s} to the starting point x_s. This framework also contains the case of a metric on a discrete graph, as detailed in Section 2.2.3, and in this case the Fast Marching algorithm is the Dijkstra algorithm.

At each step of the Fast Marching propagation, detailed in Algorithm 4, the index i of the front with minimum distance is selected

$$i \longleftarrow \operatorname*{argmin}_{k, \Sigma_k = Front} u_k.$$

Making use of a heuristic $h_k \geqslant 0$, one replaces this selection rule by

$$i \longleftarrow \operatorname*{argmin}_{k, \Sigma_k = Front} u_k + h_k.$$

Heuristics and causality. This heuristically driven selection differs from the original one detailed in Algorithm 4, which corresponds to using $h_k = 0$. For this modified Fast Marching to compute the solution of (2.3), the causality condition (2.8) should be satisfied for the new ordering, which corresponds to the requirement that for any v,

$$\forall\, x_j \in \mathrm{Neigh}(x_i), \quad \Gamma_i(v) + h_i \geqslant v_j + h_j. \tag{2.23}$$

If this condition is enforced, then the Fast Marching algorithm with a heuristically driven propagation computes the same solution on the visited points as the original Fast Marching. The advantage is that if h_i is well chosen, the number of points visited before reaching x_e is much smaller than the number of points visited by the original algorithm where $h_i = 0$.

A* algorithm on graphs. For a metric on a discrete graph, as detailed in Section 2.2.3, $u_i = d(x_s, x_i)$ is the geodesic distance on the graph between the initial point $x_s \in \Omega$ and a vertex x_i of the graph. The heuristic is said to be admissible if

$$\forall\, i \sim j, \quad h_i \leqslant W_{i,j} + h_j. \tag{2.24}$$

One can show that this condition implies the causality condition (2.23), so that the heuristically driven selection computes the correct distance function. Furthermore, defining $\tilde{W}_{i,j} = W_{i,j} + h_j - h_i \geqslant 0$, one has, for any path $\gamma \in \mathcal{P}(x_i, x_j)$, $L(\gamma) = \tilde{L}(\gamma) + h_j - h_i$, where $\tilde{L}$ is the geodesic length for the metric $\tilde{W}$. This shows that geodesics for the metric $\tilde{W}$ are the same as the ones for the metric W, and that the heuristically driven propagation corresponds to a classical propagation for the modified metric $\tilde{W}$.

A weaker admissibility condition is that the heuristic does not overestimate the remaining distance to x_e

$$0 \leqslant h_i \leqslant d(x_e, x_i), \tag{2.25}$$

in which case the method can be shown to find the correct geodesic, but the propagation needs to be modified to be able to visit several time a given point.

The modification of the Dijkstra algorithm using a heuristic that satisfies (2.25) corresponds to the A* algorithm [104, 131, 201]. This algorithm has been used a lot in artificial intelligence, where shortest paths correspond to optimal solutions to some problems, such as optimal moves in playing chess. In this setting, the graph Ω is huge, and designing good heuristics is the key to solve efficiently the problem.

Fast Marching with heuristic. The extension of the A* algorithm to the Fast Marching setting was proposed by Peyré and Cohen [218]. In this setting, where continuous geodesic distances are approximated, condition (2.24) does not implies anymore that the modified marching gives the exact same solution as the original algorithm. Numerical tests however suggest that imposing (2.24) is enough to ensure a good precision in the computation of the geodesic distance and the geodesic curve.

Efficient heuristics. An efficient heuristic should reduce as much as possible the region explored by the Fast Marching, which corresponds to

$$R = \{x_i \in \Omega \setminus d(x_s, x_i) + h_i \leqslant d(x_s, x_e)\}.$$

It means that h_i should be as large as possible, while satisfying the admissibility condition (2.24).

Condition (2.25) implies that the geodesic curve $\gamma^\star$ between x_s and x_e is contained in the explored area R. The case where $h_i = d(x_e, x_i)$ is a perfect heuristic, where R is restricted to the geodesic $\gamma^\star$, so that the front only propagates along this geodesic.

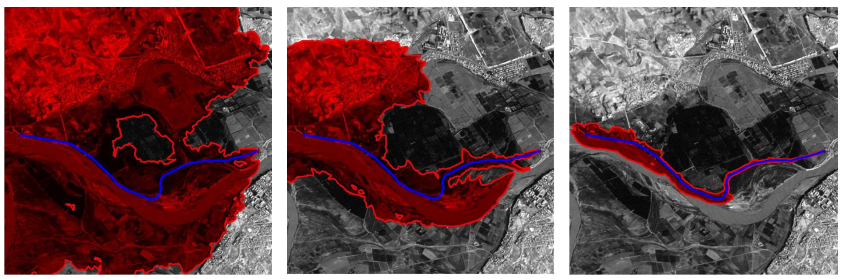
Unfortunately, this optimal heuristic is an oracle that one is not allowed to use, since computing $d(x_e, x_i)$ is as computationally difficult as the original problem of computing $d(x_s, x_i)$. One thus has to resort to sub-optimal choices.

Figure 2.13 shows for instance the effect of using a weighted oracle $h_i = \lambda d(x_e, x_i)$ for several value of λ. This shows the advantage of using

a good estimate of the (unknown) distance $d(x_e, x_i)$ since the explored region

$$R_\lambda = \{x_i \in \Omega \setminus d(x_s, x_i) + \lambda d(x_e, x_i) \leqslant d(x_s, x_e)\}$$

shrinks around the geodesic. For a Euclidean metric $T_x = \mathrm{Id}_d$ in $\mathbb{R}^d$, R_λ is an ellipsoid that shrinks along the segment $[x_s, x_e]$ when λ tends to 1.



$\lambda = 0$ $\lambda = 0.5$ $\lambda = 0.9$

Fig. 2.13 *Example of propagations with a heuristic $h_i = \lambda d(x_e, x_i)$ for various λ.*

Euclidean-based heuristics. Many strategies can be used to estimate a heuristic. For instance, for a Riemannian metric T_x, one can use a Euclidean distance approximation

$$h_i = \|x_i - x_e\|_{T_0} \quad \text{where} \quad \forall\, x, \quad T_0 \leqslant T_x, \tag{2.26}$$

where, for two symmetric positive matrices $A \leqslant B$ indicates that $\|\cdot\|_A \leqslant \|\cdot\|_B$. We note that this heuristic satisfies (2.24).

In the case of an isotropic metric $T_x = W(x)^2 \mathrm{Id}_d$, one obtains

$$h_i = \rho \|x_i - x_e\| \quad \text{where} \quad \rho = \min_{x \in \Omega} W(x).$$

This choice is popular in video games for Dijkstra propagation on graphs, and if $x_i \in \mathbb{R}^d$ and $W_{i,j} = 1$, one chooses $h_i = \|x_i - x_e\|$ which is the straight line distance.

Other geometric approaches based on a Euclidean approximation have been proposed, for instance using Euclidean clustering of the node of a graph [284].

Landmark-based heuristics. The approximation (2.26) of $d(x_e, x_i)$ by h_i can be rather crude for highly varying metrics. In order to compute more accurate heuristic, we use an expression of the geodesic distance as a minimization

$$d(x_e, x_i) = \max_{z \in \Omega} |d(x_e, z) - d(z, x_i)|.$$

which corresponds to the reversed triangular inequality.

If one restricts the minimum to a small subset of landmark points $\{z_0, \ldots, z_{K-1}\} \subset \Omega$, one can define the following heuristic

$$h_i = \max_{0 \leqslant j < K} |d(x_e, z_j) - d(z_j, x_i)|.$$

We note that this heuristic is exact, meaning that $h_i = d(x_e, x_i)$ if there is a landmark z_j so that x_e is located on the geodesic joining z_j to x_i, see Figure 2.14, left. A more realistic case is when the geodesics joining z_j and x_e to x_i are close, which happens frequently for a metric which is low along thin features such as roads, see Figure 2.14, right.

This heuristic can be computed on the fly if the following set of geodesic distances d_j has been precomputed

$$h_i = \max_{0 \leqslant j < K} |d_j(x_e) - d_j(x_i)| \quad \text{where} \quad d_j(x_i) = d(z_j, x_i). \tag{2.27}$$

We note that this heuristic satisfies (2.24).

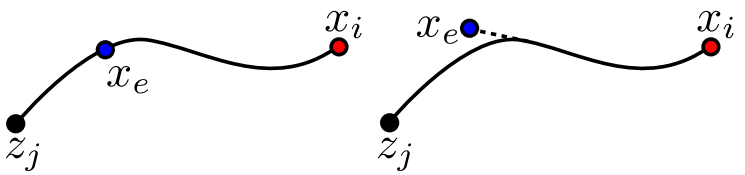


Fig. 2.14 *Left: ideal case where the heuristic is exact, $h_i = d(x_e, x_i)$. Right: case where the heuristic is good $h_i \approx d(x_e, x_i)$.*

The resulting landmark driven algorithm was originally proposed in [124] for the A* on graphs, and extended in [218] for an arbitrary discretized Riemannian manifold. The method pre-computes the distance maps d_j using K Fast Marching propagations. Then, when a query is performed to compute a minimal path between two arbitrary

points x_s, x_e, it makes use of this pre-computation to save time on the propagation by making use of the heuristic (2.27). Finding good locations $\{z_j\}_{0 \leqslant j < K}$ is a difficult problem, see [218] for a study of different approaches.

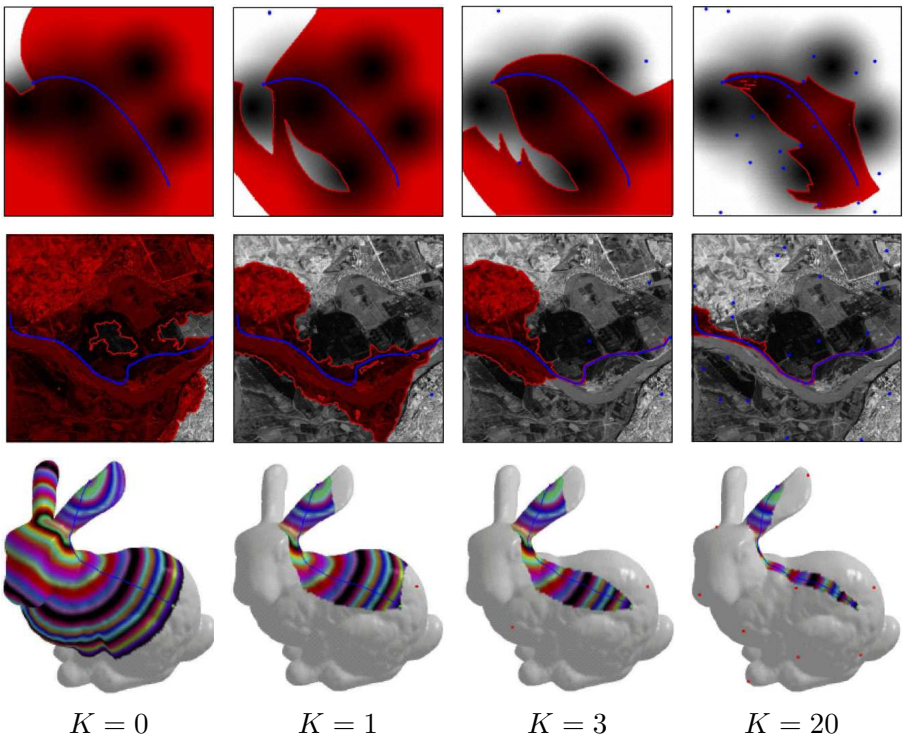


Fig. 2.15 *Heuristically driven propagations with an increasing number of landmark points, for an isotropic metric in 2D (top rows) and a 3D surface (bottom row).*

The heuristic h_i defined in (2.27) converges to the ideal heuristic $d(x_e, x_i)$ when the number K of landmarks increases. In practice, it is a trade off between pre-computation time and memory versus accuracy of the heuristic. Figure 2.15 shows the influence of the number K of landmarks on the region explored by the Fast Marching.

2.6 Computation of Voronoi Segmentation and Medial Axis

2.6.1 Geodesic Voronoi Computation

Computing Voronoi segmentation (4.1) is at the heart of segmentation methods described in Section 4.1.1, and of geodesic meshing algorithm described in Chapter 4.

Exact discrete computation. For a discrete set $S = \{x_i\}_{i \in I}$ of starting points, the Voronoi segmentation is easily computed if the whole set of distances $\{U_{x_i}\}_{i \in I}$ has been computed.

If $\mathcal{C}_i$ and $\mathcal{C}_j$ are two neighboring Voronoi cells, their common boundary is

$$\mathcal{C}_i \cap \mathcal{C}_j \subset \left\{ x \in \Omega \setminus U_{x_i}(x) = U_{x_j}(x) \right\}.$$

On a triangulated 2D domain, $U_{x_i}(x)$ is discretized as a function that is affine on each triangle. The boundary $\mathcal{C}_i \cap \mathcal{C}_j$ is thus a piecewise linear polygon, as shown on Figure 2.16, left. The location of the vertices of this polygonal boundary are found by traversing the edges and computing the intersection of 1D affine function along the edges, as shown on Figure 2.16, bottom left.

If $\mathcal{C}_i, \mathcal{C}_j$ and $\mathcal{C}_k$ are neighboring cells, the triple points where the cells intersect are

$$\mathcal{C}_i \cap \mathcal{C}_j \cap \mathcal{C}_k \subset \left\{ x \in \Omega \setminus U_{x_i}(x) = U_{x_j}(x) = U_{x_k}(x) \right\}.$$

On a triangulated surface, they are found by traversing the triangles and computing the intersection of three 2D affine functions, as shown on Figure 2.16, bottom right.

On a quadrangular 2D grid, such as for instance on an image, the computation is more complicated. The distance maps $U_{x_i}(x)$ can be represented as continuous functions using bilinear interpolation. In this case, the boundaries $\mathcal{C}_i \cap \mathcal{C}_j$ of the Voronoi cells are continuous curves composed of hyperbolas in each square of the discrete domain.

These extraction procedures extend to higher dimensional manifolds.

When using Fast Marching computation, it is possible to avoid unnecessary computation by running the propagation of the set of fronts

emanating from each $x_i \in S$ in parallel, and allowing the fronts to overlap on a depth of a few samples.

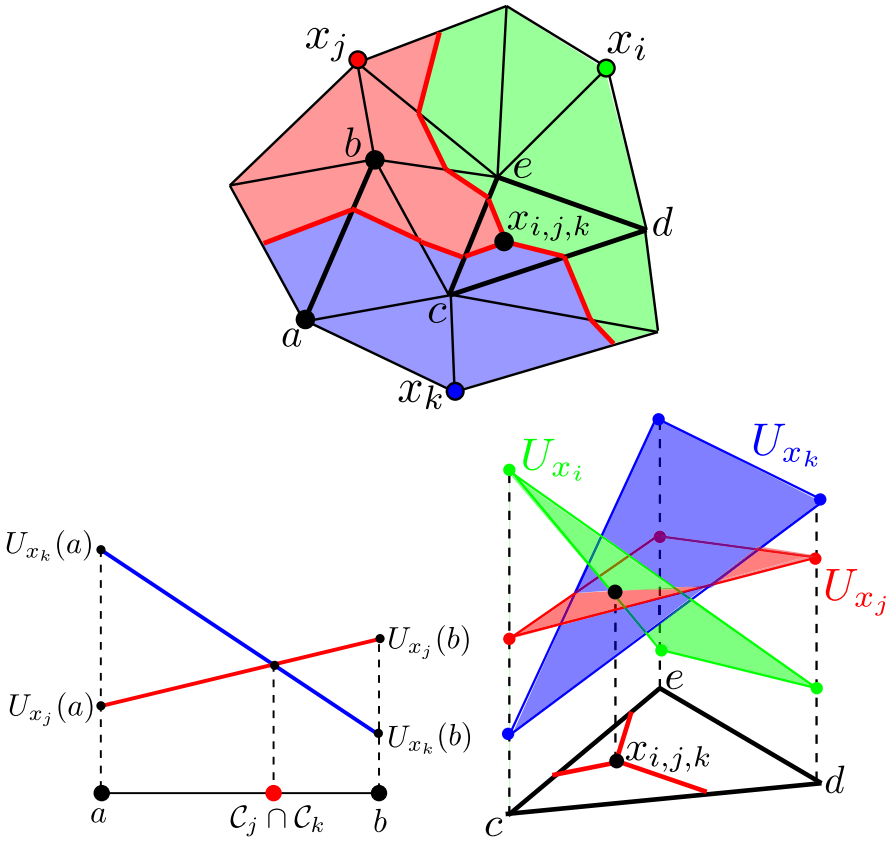


Fig. 2.16 *Top: extraction of piecewise affine Voronoi boundaries on 2D triangulations. Bottom: the geodesic distances on the edge (a, b) and triangle (c, d, e) extracted from the triangulation on the top are displayed as affine functions.*

Approximate computation. This overlapping is more difficult to implement when using a Gauss-Seidel iterative algorithm. It is possible to use a single propagation, but maintain an additional information $\ell_i \in I$ that approximates the partition function $\ell(x_i)$ defined in (1.19). This allows to retrieve after the propagation an approximate partition

function. This computation is however approximate and does not give an exact discrete Voronoi segmentation. The partition can however be used afterward as an indicator of the locations where the fronts are allowed to overlap to implement a parallel propagation.

Each time the update operator

$$u_i \leftarrow \Gamma_i(u)$$

is applied at a point x_i, by either an iterative or a fast marching algorithm, one also applies an update operator to compute ℓ_i from its neighbors,

$$\ell_i \leftarrow \tilde{\Gamma}_i(u, \ell).$$

This update is computed by assigning the index of the closest neighboring point used to perform the update. More precisely, if

$$\Gamma_i(u) = v_{i,j,k} \quad \text{where} \quad t_{i,j,k} \in \text{Neigh}(x_i)$$

where $v_{i,j,k}$ is defined in (2.5), one defines

$$\tilde{\Gamma}_i(u, \ell) = \begin{cases} \ell_j & \text{if} \quad |v_{i,j,k} - u_j| < |v_{i,j,k} - u_k|, \\ \ell_k & \text{otherwise.} \end{cases} \tag{2.28}$$

Figure 2.17 shows examples of Voronoi cells on a surface embedded in $\mathbb{R}^3$.

2.6.2 Shape Skeletons Computation

For a 2D manifold, if S is a smooth closed curve $c(t), t \in [0, 1]$, then $\text{MedAxis}(S)$ is a connected union of 1D curves. If $c(t)$ is the boundary of a 2D object, the medial axis is often referred to as the skeleton of the shape.

As proposed in [269], it is possible to approximate the medial axis of a smooth curve $c(t)$ by processing the nearest neighbor index map $\ell(x)$, see also [132, 278]. In this setting,

$$S = \{x_i = c(i/K)\}_{i=0}^{K-1}$$

is assumed to be a dense sampling of a smooth curve. The singularity points of the distance function U_S are difficult to detect from the

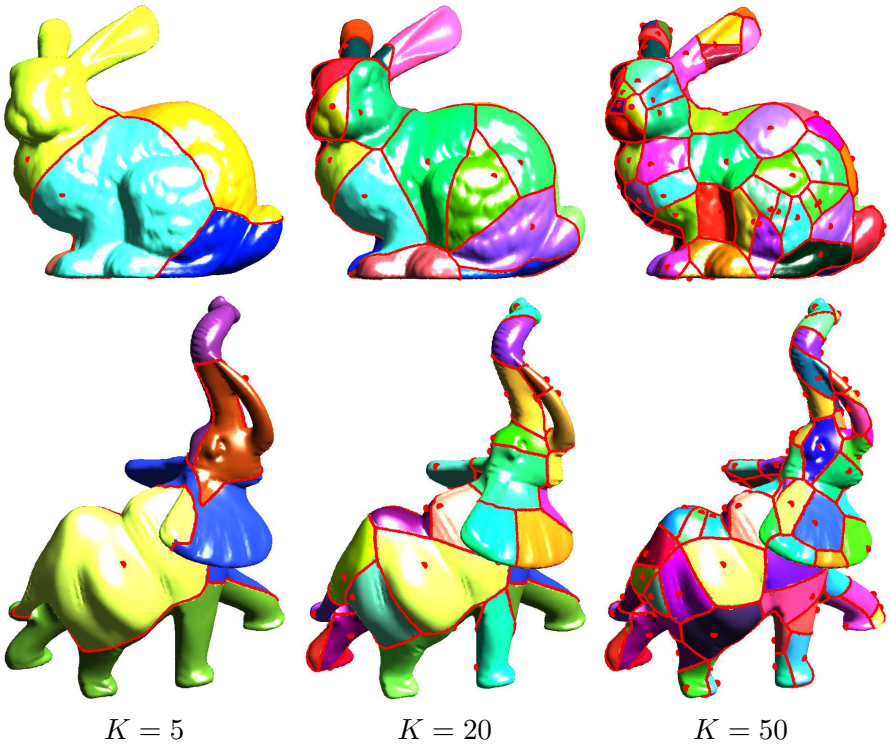


Fig. 2.17 *Example of Voronoi segmentations $\mathcal{V}(S)$ for an increasing number of seeding points.*

variation of U_S. These singularity points are, however, located approximately at sharp transition of the partition function $\ell(x)$. In the following, we assume that $\ell(x) \in \{0, \ldots, K-1\}$ indicates the index $x_{\ell(x)} \in S$ of the closest point, and not the closest point itself.

They can be detected by computing the magnitude map of the gradient $\|\nabla \ell(x)\|$, that is computed numerically on the discrete grid by finite differences from $\{\ell_i\}_i$. One should be careful about the fact that the derivative should be estimated using finite differences modulo K where $K = |S|$ is the number of sampling points along the curve, so that $-K/2 \leqslant \nabla \ell(x) \leqslant K/2$. This is because $\ell(x) \in \{0, \ldots, K-1\}$ exhibits an artificial jump discontinuity when ℓ passes from $K-1$ to 0.

The medial axis can then be approximated by thresholding the mag-

nitude of the gradient

$$\mathrm{MedAxis}_\tau = \{x \in \Omega \setminus \|\nabla \ell(x)\| \geqslant \tau\}\,.$$

Increasing the value of τ regularizes the medial axis. Figure 2.18 shows examples of such approximated medial axis for a couple of values of τ.

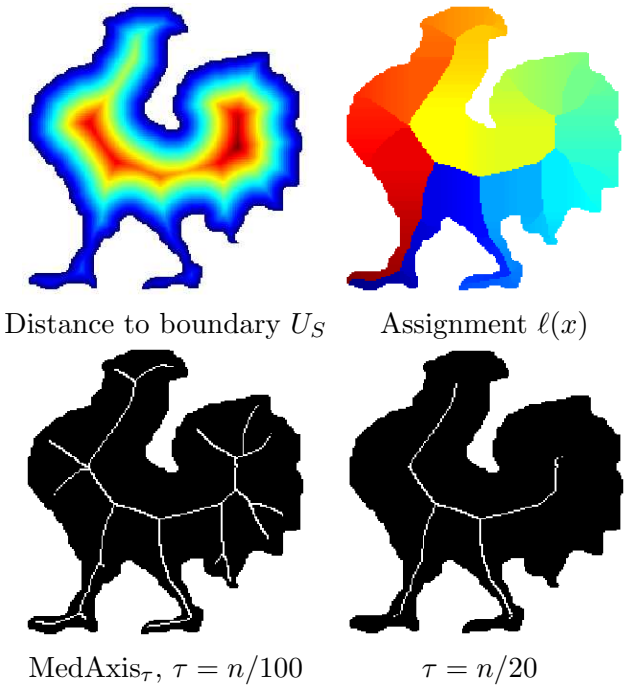


Fig. 2.18 *Computation of the approximated medial axis, for an image of $n \times n$ pixels.*

Other methods to compute skeletons. There exists a variety of alternative fast methods to compute skeletons and regularize their geometry for the Euclidean distance $T_x = \mathrm{Id}_2$. A subset of the edges of the Voronoi diagram of a dense sampling of a curve was originally proposed in [203]. It can be shown to approximate the skeleton of the continuous curve. One can also use curves evolving according to PDEs similar to active contours [167, 145] or deformable sets of disks [301].

There have also been several attempts to give alternate formulation

for the skeleton that is more meaningful for shape recognition [300] or to prune potentially noisy skeletons [252].

2.7 Distance Transform

In the special case where $\Omega = [0,1]^2$ equipped with a uniform metric $T_x = \mathrm{Id}_2$, U_S is called distance transform (DT) of the binary shape S. Compared to the general Riemannian case, taking into account the special structure of this problem allows to design several exact or approximate fast algorithms to compute DT. A flourishing literature exists on this subject – and we follow in this section the classification of the review [114].

2.7.1 Propagation algorithms

Distance transform over a continuous domain. In DT framework, since the domain Ω is convex, the geodesic distance is the Euclidean distance

$$\forall\, (x,y) \in \Omega^2, \quad d(x,y) = \|x-y\|.$$

The knowledge of the partition function,

$$\ell(x) = \operatorname*{argmin}_{y \in S} \|x-y\| \tag{2.29}$$

already defined in (1.19), allows to compute the distance map to the starting point

$$U_S(x) = \|x - \ell(x)\|. \tag{2.30}$$

A central idea to compute the distance map U_S is thus to keep track of the partition function $\ell(x) \in S$ using an iterative algorithm [194].

In a continuous setting, it is possible to show that for any point $x \in \Omega \backslash S$, there exists a neighboring point with the same closest point in S

$$\exists y \in B(x), \quad \ell(y) = \ell(x), \tag{2.31}$$

where the neighborhood $B(x)$ is defined in Section 2.1.1. This point can be chosen at the intersection of $\gamma^\star$ and $\partial B(x)$.

We now show how this setting can be discretized. Notice that unlike in the frameworks developed in the previous sections, it is possible to

compute exact discrete solutions to the distance transform problem in extremely low computational time.

Discrete update of the partition function. The distance function U_S is discretized using an uniform grid, and is represented as an image of $N = n \times n$ pixels. The set of pixels should be understood as a graph where each pixel x_i is connected to each of its neighbors in $\text{Neigh}(x_i)$, which is written as $j \sim i$, similarly to the graph setting described in Section 2.2.3.

The partition function $\ell(x) \in S$ is approximated at a point x_i of the grid as $\ell_i \in S$, and ℓ_i is computed in parallel to a front propagation algorithm. The steps of the algorithm are the same as the Dijkstra and Fast Marching methods, detailed in Algorithm 4. The difference comes from the update of the distance map, since one first applies an update of the partition function

$$\ell_i \leftarrow \tilde{\Gamma}_i(\ell) \tag{2.32}$$

before updating the distance according to (2.30)

$$u_i \leftarrow \Gamma_i(u) = \|x_i - x_{\ell_i}\|.$$

The discrete counterpart of the continuous property (2.31) reads

$$\forall\, x_i \in \Omega \;\; \exists x_j \in \text{Neigh}(x_i) \;\; \ell_i = \ell_j. \tag{2.33}$$

The update of the partition function (2.32) is thus defined as

$$\tilde{\Gamma}_i(\ell) = \ell_{j^\star} \quad \text{where} \quad j^\star = \operatorname*{argmin}_{j \sim i} \|x_i - \ell_j\|.$$

This is similar to the partition function update (2.28) defined for the Fast Marching.

Note that since only roots of integer values are computed during the execution of the algorithm, several improvements of the implementation can be performed, as detailed in [114].

Most unfortunately, as shown in [83], (2.33) is only approximate, and does not allow for an exact computation of the DT, whatever the size of $\text{Neigh}(x_i)$ is. A second pass can be used with a larger $\text{Neigh}(x_i)$ in order to correct ℓ_i at points where it might be wrong after the first pass [85].

2.7.2 Raster scan methods

Since the computational bottleneck in propagation algorithms is the maintenance of a bucket structure and the selection of the point to update, methods have been proposed in which the points are updated in some predefined order – using sets of ad-hoc update operators.

The reasoning of the previous section applies here, and one cannot expect to have an exact algorithm if the update operators use neighborhood of fixed size.

Popular raster scan methods are the 4SED and 8SED methods [86], which uses 4 scans of the image, and update the points using respectively neighborhoods $\text{Neigh}(x_i)$ of maximal size 4 and 8 – notice that the neighborhood are different from one scan to another. Algorithm 5 details the 4SED algorithm, and Figure 2.19 displays results obtained with the 4SED method.

While its complexity is linear, this algorithm does not lead to an exact value of the DT. It can be corrected efficiently by a post-processing step, leading to a linear time exact algorithm [84].

More recently, exact algorithms with only two raster scans and an adaptive neighborhood $\text{Neigh}(x_i)$ was proposed [260, 261].

Algorithm 5: 4SED-algorithm.

$\forall (i,j) \in [0, n-1]^2$, set : $\text{Neigh}_1(i,j) = \{(i-1,j),(i,j-1)\}$, $\text{Neigh}_2(i,j) = \{(i+1,j)\}$, $\text{Neigh}_3(i,j) = \{(i+1,j),(i,j+1)\}$, $\text{Neigh}_4(i,j) = \{(i-1,j)\}$.

for i *from* 0 *to* $n-1$ **do**
  for j *from* 0 *to* $n-1$ **do**
    Update (i,j) using Neigh_1 (**scan 1**)
  for j *from* $n-1$ *to* 0 **do**
    Update (i,j) using Neigh_2 (**scan 2**)

for i *from* $n-1$ *to* 0 **do**
  for j *from* $n-1$ *to* 0 **do**
    Update (i,j) using Neigh_3 (**scan 3**)
  for j *from* 0 *to* $n-1$ **do**
    Update (i,j) using Neigh_4 (**scan 4**)

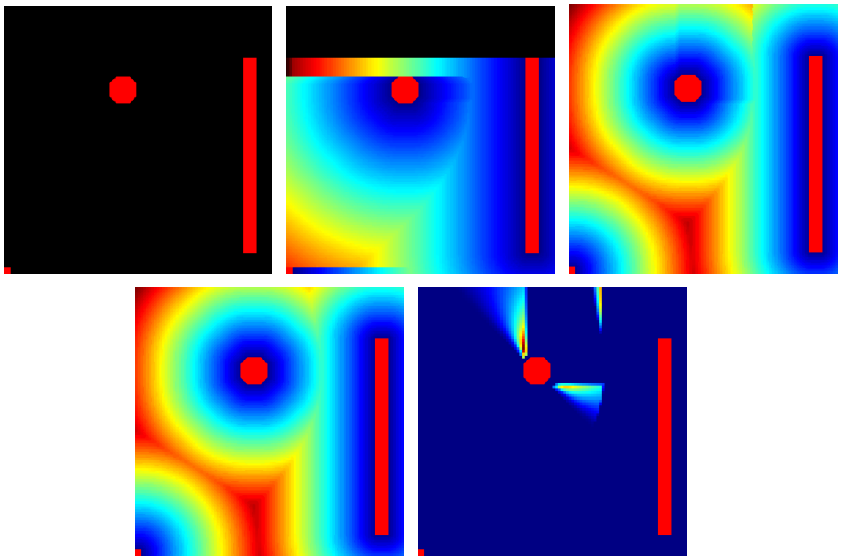


Fig. 2.19 *4SED method. Top row, from left to right : Starting set S, consisting of 3 different areas. Distance map after scans 1 and 2. Notice that distance information is only propagated in the bottom direction, and that some areas still have an ∞ value. Distance map after scans 3 and 4. Bottom row, from left to right : Ground Truth distance map, and relative error of the 4SED method. The computed distance map is correct on the major part of the image, but relative errors of $\sim 27\%$ occurs in some areas.*

2.7.3 Independent scan methods

Independent scanning methods are an attempt to speed up the DT computation by using a dynamic programming like approach.

The main idea is to first compute the distance to the closest point in the same row of the image :

$$\ell'_i = \underset{\substack{x_j \in S \\ (x_j)_1 = (x_i)_1}}{\operatorname{argmin}} |x_i - x_j|. \tag{2.34}$$

ℓ' can be computed on each line independently, which makes its computation extremely efficient [238].

In a second step, ℓ is computed from ℓ'. Several strategies are available to perform this operation, and we again refer the interested reader to the review [114] and to the seminal articles [246, 184, 182].

Notice that this framework can be generalized for k-dimensional images. Independent DT computations are computed recursively on

$(k-1)$-dimensional slices of the original images, and put together to obtain the full DT transform of the original image.

2.8 Other Methods to Compute Geodesic Distances

In this section, we detail some other methods which allow to compute distance maps in different settings.

2.8.1 Mathematical Morphology

Mathematical morphology can be used to compute approximations of geodesic distance maps.

In the framework of mathematical morphology, a shape is represented as subset S of Ω. If B_r denotes a small disk of radius r centered at the origin, the dilation of S by the disk B_r is ([253])

$$\delta_{B_r} S = \{x + b \in \Omega \mid x \in S, b \in B_r\}. \tag{2.35}$$

If B_{rx} furthermore denotes the same disk of radius r centered at point x, it is possible to equivalently define

$$\delta_{B_r} S = \bigcup_{x \in S} \{B_{rx}\}. \tag{2.36}$$

Assuming a uniform isotropic potential over Ω, one can show that if Ω is convex, then $\{x \in \Omega,\ U_S(x) \leqslant r\} = \delta_{B_r} S$. Thus the iso-distance curve $\{x \in \Omega,\ U_S(x) = r\}$ can be computed as the border of $\delta_{B_r} S$.

Equation (2.36) suggests a method to compute this set. For any point in S, one can mark all the points in $\delta_{B_{rx}}$, and find the border of the resulting set.

Numerically, it is possible to define a discrete counter part of the dilation operator (Figure 2.20). As an example, assuming that Ω is discretized on a regular grid, we denote B_d the discrete counter-part of B, and by S_d the one of S. The dilation is then defined in the same way as in the continuous case $\delta_{B_d} S_d = \{x + b \in \Omega \mid x \in S_d, b \in B_d\}$.

It is then possible to use the method described in Algorithm 6 to find an approximation M of $\{x \in \Omega,\ U_S(x) \leqslant r\}$ in linear time.

This method however does not work if Ω is not convex. In this case, the concept of geodesic dilation was introduced in [248] and [247]. It

Fig. 2.20 *Discrete dilation of the black shape S_d by the red kernel B_d.*

consists in several small dilations. For a small ball B_r, we define

$$\delta_{B_r}^n S = \underbrace{\delta_{B_r} \dots \delta_{B_r}}_{n \text{ times}} S. \tag{2.37}$$

Again, a corresponding discrete operator B_d can be defined (see Figure 2.21).

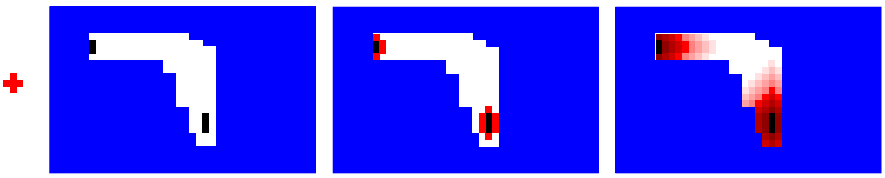


Fig. 2.21 *Geodesic dilation of the black shape by the red kernel (left). The first iteration is shown on the middle figure, while several others iterations of this dilation are shown on the right figure.*

This operator is applied iteratively, thus leading to a shape $\delta_{B,\Omega}^n S$ which approximates the set of point $x \in \Omega$ such that $U_S(x) \leqslant nr$. Note that computing $\delta_{B,\Omega}^n S$ is equivalent to performing a breadth-first search algorithm on the discretization of the manifold, where the edges are given by $x \sim y \Leftrightarrow x - y \in B_d$. If n points are visited, the methods thus run in $\mathrm{O}(n)$ time, where the constant depends on the size of B_d.

This approximation is subject to a metrication error due to the shape of B_d. As an example, the distance computed in Figure 2.21,

Algorithm 6: Morphologic dilation computation.

Initialization:
$\forall\, x_i \in \Omega,\ M_i \leftarrow 0$
for $x_i \in S$ **do**
  for $x_l \in \Omega \backslash S$ *such that* $\|x_i - x_l\| \leqslant r$ **do**
    $M_l \leftarrow 1$

right, is indeed the Manhattan distance d_1. This can be improved (but not completely dealt with) by taking a bigger B_d, at the sake of a possible loss of the topology of Ω, and a loss of precision in the areas where the boundary of Ω is concave.

Voronoi diagrams for this morphological distance give rise to the celebrated watershed segmentation method, see Section 4.1.1.

2.8.2 Geodesics on Weighted Region

Assuming that Ω is partitioned into a family polyhedral subsets $\Omega_1 \dots \Omega_n$ such that $W(x) = W_i$ is constant over Ω_i, exact geodesics can be computed [192].

This framework was extended when translational flow is present in the polyhedra [234].

2.8.3 Exact Discrete Geodesics on Triangulations

An alternative way to approximate geodesic distances on surfaces is to compute the exact distance on a triangulated mesh approximating a smooth surface. The fastest algorithm run in $O(N \log(N))$ on a surface with N vertices [59]. A simpler approach [191] computes the distance to a single starting point in $O(N^2)$ operations by propagating intervals over which the distance is linear. This approach can be accelerated by performing approximate computations [264]. See also [290, 3, 226, 225] for other approaches.

Computing locally minimizing curves (that might not be globally shortest paths) on such triangulated mesh is obtained by path tracing methods [224].

2.9 Optimization of Geodesic Distance with Respect to the Metric

In some applications, the object of interest is not the geodesic distance, but rather the metric itself. This includes landscape design problems, where the metric represents the ground height to optimize, or seismic inverse problems, where the metric models the unknown ground velocity [29].

For the sake of simplicity, we detail here the case of an isotropic metric $T_x = W(x)^2 \mathrm{Id}_d$, but the algorithm extends to general manifolds for which Fast Marching methods can be used.

2.9.1 Sub-gradient of the Geodesic Distance

In many applications, one is facing the problem of computing an unknown metric $W(x)$ that optimizes a variational energy that depends on the geodesic distance $d_W(x_s, x_e)$ between pairs of points $x_s, x_e \in \Omega$, where we have made the dependency on the metric W explicit. A basic ingredient to solve such a problem is the gradient of the geodesic distance $d_W(x_s, x_e)$ with respect to the metric W. This gradient can be formally derived as

$$d_{W+\varepsilon Z}(x_s, x_e) = d_W(x_s, x_e) + \varepsilon \langle Z, \nabla d_W(x_s, x_e) \rangle + o(\varepsilon) \qquad (2.38)$$

where

$$\nabla d_W(x_s, x_e) : \Omega \to \mathbb{R}$$

is the gradient mapping. Note that for each (x_s, x_e) one obtains a different mapping.

The expansion (2.38) is only formal, since the mapping $W \mapsto d_W(x_s, x_e)$ is not necessarily smooth, and in particular it is not differentiable if x_s and x_e are connected by several geodesics. However, since $d_W(x_s, x_e)$ is the minimum of all paths lengths $L(\gamma)$ for $\gamma \in \mathcal{P}(x_s, x_e)$, it is the minimum of linear functions of W, so d_W is a concave function of W. It is thus possible to interpret (2.38) as $\nabla d_W(x_s, x_e)$ being a super-gradient of the geodesic distance. Since the term "super-gradient" of a concave functional is not very well used, we refer to it as a "sub-gradient" which is the term used for convex functionals.

A formal derivation shows that if $\gamma^\star \in \mathcal{P}(x_s, x_e)$ is the unique geodesic path between x_i and x_j, then

$$\langle Z, \nabla d_W(x_s, x_e) \rangle = \int_0^1 Z(\gamma^\star(t)) \mathrm{d}t,$$

so that $\nabla d_W(x_s, x_e)$ is in fact a 1D measure supported along the geodesic curve. Computing $\nabla d_W(x_s, x_e)$ directly from this continu-

ous definition is thus difficult. A better option is to compute the sub-gradient of a discrete geodesic distance, which is well-posed numerically and can be obtained with a fast algorithm.

2.9.2 Sub-gradient Marching Algorithm

We consider the discrete Eikonal equation (2.3) that defines a discrete geodesic distance. Since we consider here the special case of an isotropic metric on a regular grid, the equation can be equivalently written using up-wind finite differences (2.16).

This Sub-gradient Marching algorithm that we detail next was introduced in [22], where a proof of the validity of the method is given. It is applied in [56] to a variational traffic congestion problem [286], where the discrete sub-gradient is required to obtain a convergence of the numerical method.

Sub-gradient to a starting point. We aim at computing all the sub-gradients to a given starting point x_s

$$\nabla u_i = \nabla d_W(x_s, x_i) \in \mathbb{R}^N$$

with a method similar to the Fast Marching algorithm, detailed in Algorithm 4, that computes all the distances

$$u_i = d_W(x_s, x_i) \in \mathbb{R}.$$

Note that the dependency on both W and x_s has been made implicit.

Sub-gradient marching algorithm. The Fast Marching algorithm iteratively makes use of the update operator

$$u_i \leftarrow \Gamma_i(u),$$

whose values are computed by solving a quadratic equation. The sub-gradient marching algorithm makes use of a similar update step,

$$\nabla u_i \leftarrow \Gamma_i^{\nabla}(\nabla u),$$

that is applied to the sub-gradient map each time Γ_i is applied to the distance map.

The resulting algorithm has the exact same structure as the original Fast Marching propagation detailed in Algorithm 4. Since the manipulated gradients ∇u_i are vector of $\mathbb{R}^N$, the overall complexity of computing all the vectors ∇u_i is $O(N^2 \log(N))$.

Figure 2.22 shows examples of discrete sub-gradients computed with this algorithm. Note how the gradient for the constant metric is supported near the segment joining the two point. Note also how the support of the gradient split for a varying metric, when (x_s, x_e) are close to a configuration where two distinct geodesics exist between the two points.

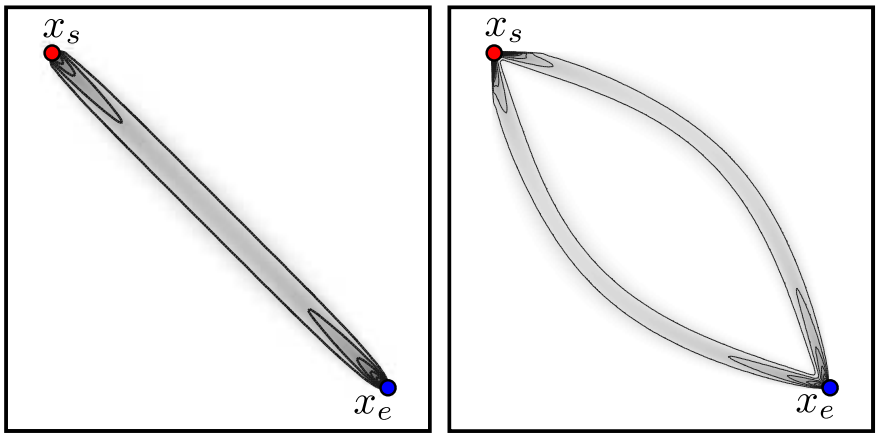


Fig. 2.22 *Example of sub-gradient $\nabla d_W(x_s, x_e)$ computed for a constant (on the left) and a varying (on the right metric which is large in the middle of the image.*

Sub-gradient update step. We denote as x_j and x_k the two adjacent points of i that support the update in (2.10), which means that

$$\Gamma_i(u) = v_{i,j,k}$$

where $v = v_{i,j,k}$ is defined in (2.15) as the solution of

$$(v - u_j)^2 + (v - u_k) = \varepsilon^2 W_i^2. \tag{2.39}$$

We detail here the case where the quadratic equation has two solutions. In this case, the updated gradient $\nabla v = \Gamma_i^{\nabla}(\nabla u)$ is obtained by

differentiating (2.39) with respect to the metric W, which gives

$$\alpha_j(\nabla v - \nabla u_j) + \alpha_k(\nabla v - \nabla u_k) = \varepsilon^2 W_i \delta_i \quad \text{where} \quad \left\{ \begin{array}{l} \alpha_j = u_i - u_j \\ \alpha_k = u_i - u_k \end{array} \right. ,$$

and where $\delta_i \in \mathbb{R}^N$ is defined as $\delta_i(j) = 0$ if $j \neq 0$ and $\delta_i(i) = 1$. One thus obtains the definition of the sub-gradient update

$$\Gamma_i^{\nabla}(\nabla u) = \frac{1}{\alpha_j + \alpha_j} \left(\varepsilon^2 W_i \delta_i + \alpha_j \nabla u_j + \alpha_k \nabla u_k \right).$$

This algorithm has been extended to metrics on 3D meshes [98].

2.9.3 Inverse Problems Involving Geodesic Distances

The sub-gradient marching method has been applied to various convex and non-convex inverse problems involving geodesic distance, see [56, 22] for example of isotropic metric, and [98] for inverse problems on 3D meshes.

As an example, one can consider a simple convex problem of landscape design, where the metric is

$$\max_{W \in \mathcal{W}} \sum_{(i,j) \in \mathcal{D}} d_W(y_i, y_j) \tag{2.40}$$

where $\{y_i\}_{i \in I}$ is a set of landmarks, $\mathcal{D}$ is a set of connections between pairs of points and where $\mathcal{W}$ is a set of convex constraints, for instance

$$\mathcal{W} = \left\{ W \setminus \forall\, i,\ W_{\min} \leqslant W_i \leqslant W_{\max} \quad \text{and} \quad \sum_i W_i = 1 \right\}.$$

This can be interpreted as designing a ground elevation, with constraints on the total available material, and on the minimum and maximum height of the landscape, so that locations (x_i, x_j) for $(i, j) \in \mathcal{D}$ are maximally distant one from each other.

Existence and uniqueness of a solution of the continuous problem is investigated in [54]. A projected gradient descent method, detailed in [22], approximates the solution of (2.40) iteratively

$$W^{(k+1)} = \mathrm{Proj}_{\mathcal{W}} \left(W^{(k)} + \eta_k \sum_{(i,j) \in \mathcal{D}} \nabla d_{W^{(k)}}(y_i, y_j) \right)$$

where $\mathrm{Proj}_{\mathcal{W}}$ is the orthogonal projection on $\mathcal{W}$, and $\eta_k \approx 1/k$ are the gradient step size. The sub-gradient $\nabla d_{W^{(k)}}(y_i, y_j)$ are computed at each iteration using the sub-gradient marching algorithm.

Figure 2.23 shows an example of optimal metric computed with this method.

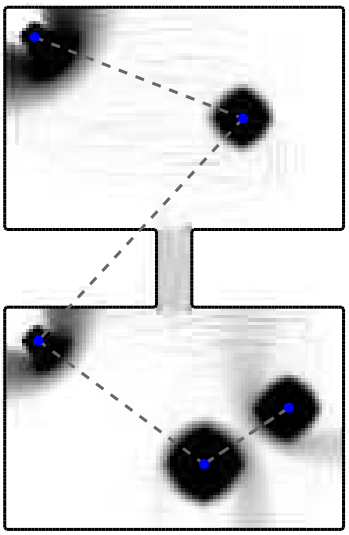


Fig. 2.23 *Example of optimal metric solving* (2.40)*, where the connections* $(i, j) \in \mathcal{D}$ *are shown in dashed lines.*

3

Geodesic Segmentation

A major area of applications of geodesic methods in image processing is to detect curvilinear features and perform segmentation. The metric can be designed for geodesic curves to follow the edges and tubular structures, or, on the contrary, for geodesic balls to stop near features. These two points of view for using Fast Marching were applied to active contours, the first one to obtain a global minimum as a minimal path [70], the second one using the front of the Fast Marching propagation to compute a curve evolution [178].

3.1 From Active Contours to Minimal Paths

3.1.1 Snakes and Geodesic Active Contours

Variational curve modeling. Active contours is a class of segmentation methods that detect an object by evolving a curve according to both a smoothness constraint and a data fitting constraint. This curve is attracted by the features in the image – typically edges. These deformable models or active contours were introduced with the snakes of Kass et al. [143]

A general framework for the evolution of the active contour is the

minimization of a variational energy over curves $\gamma : [0,1] \to [0,1]^2$

$$E(\gamma) = L(\gamma) + \lambda R(\gamma) \quad \text{where} \quad L(\gamma) = \int_0^1 W(\gamma(t)) \|\gamma'(t)\| \mathrm{d}t, \quad (3.1)$$

where $R(\gamma)$ is some smoothness enforcing criterion, and $L(\gamma)$ is a data fitting energy that takes into account the features of the image through the potential W. This potential should be low near the boundary of the object to segment. Several strategies to design W are detailed in Section 3.2. λ is a non-negative real value which sets the relative importance of the two terms.

One can consider open curves that join two points x_s, x_e, and add to the minimization (3.1) the following boundary constraints :

$$\gamma(0) = x_s \quad \text{and} \quad \gamma(1) = x_e, \quad (3.2)$$

which correspond to the constraint $\gamma \in \mathcal{P}(x_s, x_e)$ as defined in (1.3). One can also consider closed curves by imposing $\gamma(0) = \gamma(1)$, in which case the minimization (3.1) is unconstrained but the derivatives with respect to t are computed modulo 1.

One can note that $L(\gamma)$ is the geodesic length of the curve according to the isotropic Riemannian metric W, as already defined in (1.2).

In the original snakes [143], the energy takes into account both the length of the curve and its bending using first and second order derivatives with respect to t, with

$$R(\gamma) = \int_0^1 \|\gamma'(t)\| + \mu \|\gamma''(t)\| \mathrm{d}t.$$

This energy is however not intrinsic to the curve geometry, since it also depends on the parameterization of the curve. This is why these two terms were replaced by length element and curvature to obtain an intrinsic energy and define a geometric model [57]. Since it is complex to deal with the curvature term, it was removed in that model, as well as in the level set approach of Malladi et al. [179]. Indeed, the Euclidean length of a curve can be used as regularization term, as can be seen in the Mumford-Shah energy [197], where penalty on the length of boundaries leads to their regularization. Regularization properties of minimal geodesics were proposed in [71] where it was noticed that the

length term could be included in the potential term and lead to the same energy as geodesic active contour [58]. If one uses $\mu = 0$ and replaces W by $W + \lambda$, the energy is restricted to the geodesic length

$$E(\gamma) = L(\gamma) = \int_0^1 W(\gamma(t))\|\gamma'(t)\|\mathrm{d}t, \tag{3.3}$$

thus defining the geodesic active contour.

Parametric curve evolution. Curve evolution corresponds to the minimization of the energy $E(\gamma_t)$ by evolving a family of curves γ_t indexed by $t \geqslant 0$. For an intrinsic energy that only depends on the geometry of the curve and not its parameterization, this minimization is governed by a partial differential equation where γ_s evolves in the direction normal to the curve

$$\frac{d}{dt}\gamma_t(s) = \beta(\gamma_t(s), n_t(s), \kappa_t(s))n_t(s), \tag{3.4}$$

where $\beta(x, n, \kappa) \in \mathbb{R}$ is the velocity, and where the outward unit normal to the curve $n_t(s)$ and the curvature $\kappa_t(s)$ at point $\gamma_t(s)$ are defined as

$$n_t(s) = \frac{\gamma_t''(s) - \langle \gamma_t''(s),\, T\rangle T}{\|\gamma_t''(s) - \langle \gamma_t''(s),\, T\rangle T\|}, \quad \text{and} \quad \kappa_t(s) = \langle n_t'(s),\, \gamma_t'(s)\rangle \frac{1}{\|\gamma_t'(s)\|^2}, \tag{3.5}$$

where

$$T = \frac{\gamma_t'(s)}{\|\gamma_t'(s)\|}. \tag{3.6}$$

One should also add the constraint (3.2) to this PDE in the case of open curves. Figure 3.1 shows a schematic display of the evolution.

For the geodesic active contour minimization of (3.3), the minimization of the weighted length $L(\gamma_t)$ leads to normal velocity

$$\beta(x, n, \kappa) = W(x)\kappa - \langle \nabla W(x),\, n\rangle. \tag{3.7}$$

For a constant metric $W = 1$, one recovers the mean-curvature motion $\beta(x, n, \kappa) = \kappa$, that corresponds to the flow that minimizes the Euclidean length of the curve. Figure 3.2, left, shows an example of mean curvature motion. Figure 3.2, right, shows an evolution toward a noisy circle on which the metric is low.

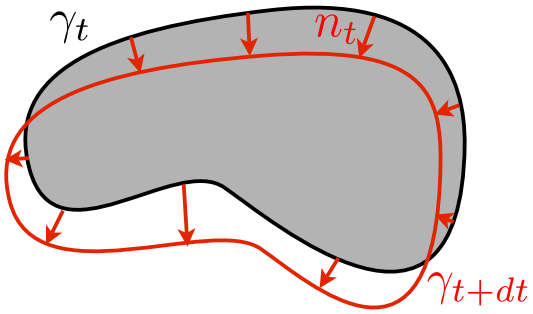


Fig. 3.1 *Curve evolution in the normal direction.*

As proposed initially in [143], evolution (3.4) can be solved by finite differences to evaluate numerically the derivatives with respect to t and s. Explicit time integration is fast but unstable, so that small time steps are required. One can use implicit time stepping, which requires the solution of a sparse linear system at each time step, and is more stable, see [69].

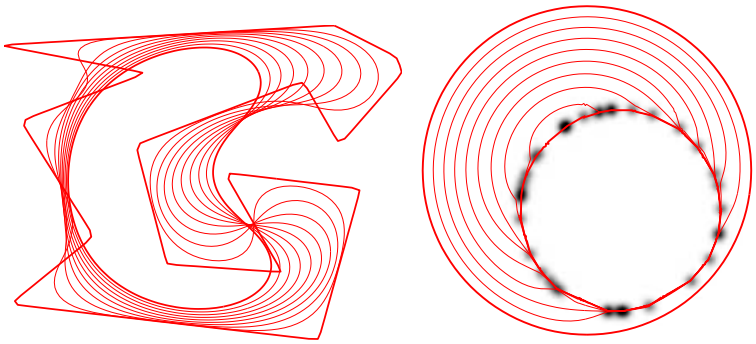


Fig. 3.2 *Left: mean curvature motion starting from a polygonal curve. Right: geodesic active contours for a metric small on a noisy circle.*

Implicit curve evolution. The curve evolution (3.7) can also be solved numerically using the level set framework of Osher and Sethian [207]. A closed curve is represented as the zero level set of a function

$\varphi_t : \mathbb{R}^2 \to \mathbb{R}$ such that

$$\{\gamma_t(s) \setminus s \in [0,1]\} = \left\{x \in \mathbb{R}^2 \setminus \varphi_t(x) = 0\right\},$$

and the interior of the domain represented by the curve is $\left\{x \in \mathbb{R}^2 \setminus \varphi_t(x) \leqslant 0\right\}$.

In this framework, union and intersection operations of two shapes represented by φ_t^1 and φ_t^2 are easily performed with algebraic manipulations

$$\varphi_t(x) = \min(\varphi_t^1(x), \varphi_t^2(x)) \quad \text{and} \quad \varphi_t(x) = \max(\varphi_t^1(x), \varphi_t^2(x)).$$

Figure 3.3 shows examples of curve embeddings using level sets.

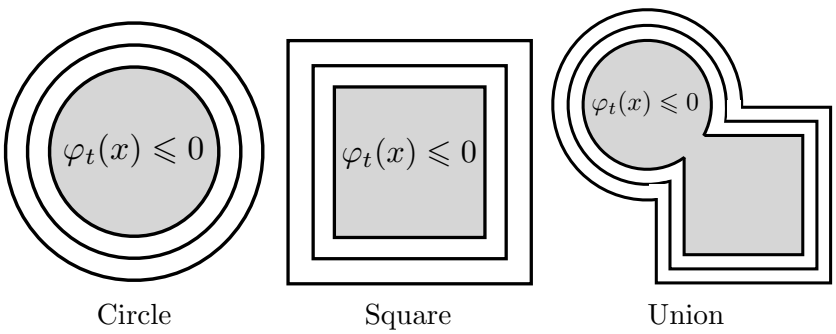


Fig. 3.3 *Example of shape embedding using level sets.*

For an arbitrary simple closed curve, a canonical choice is the signed distance function

$$\varphi_t(x) = \sigma(x)\tilde{\varphi}_t(x) \quad \text{where} \quad \tilde{\varphi}_t(x) = \min_{s \in [0,1]} \|x - \gamma(s)\| \tag{3.8}$$

where the sign $\sigma(x) = +1$ outside the domain bounded by the curve, and $\sigma(x) = -1$ inside. The unsigned distance function $\tilde{\varphi}_s$ is the unique viscosity solution of the Eikonal equation

$$\|\nabla \tilde{\varphi}_t(x)\| = 1 \quad \text{and} \quad \forall\, s \in [0,1],\ \tilde{\varphi}_t(\gamma(s)) = 0. \tag{3.9}$$

This equation can be solved in $O(N \log(N))$ operations on a regular grid of N pixels using the Fast Marching algorithm detailed in Section 2.3.

The level set implementation has the advantage of allowing merging and splitting of the curve. This enables the segmentation of several objects at the same time, which is not possible with the parametric formulation (3.7).

Within this framework, the curve evolution (3.4) becomes a PDE on the embedding function φ_s, where all the level sets (including the zero level set representing γ_s) evolve together

$$\frac{d}{dt}\varphi_t(x) = \|\nabla\varphi_t(x)\|\beta\left(\varphi_t(x), \frac{\nabla\varphi_t(x)}{\|\nabla\varphi_t(x)\|}, \mathrm{div}\left(\frac{\nabla\varphi_t}{\|\nabla\varphi_t\|}\right)(x)\right).$$

For the geodesic active contour, the level set PDE is thus

$$\frac{d}{dt}\varphi_t = \|\nabla\varphi_t\|\mathrm{div}\left(W\frac{\nabla\varphi_t}{\|\nabla\varphi_t\|}\right).$$

As the PDE evolves, the function φ_t might become unstable and exhibit large gradients. To avoid these numerical instabilities, it is necessary to enforce that φ_t is a distance function as defined in (3.8) for some values of t during the evolution. This necessitates to solve the Eikonal equation (3.9) from time to time during the level set evolution to restart the embedding function.

Local minimizer of the weighted length. Figure 3.4 shows geodesic active contour evolutions for open and closed curves. The potential W is computed using the gradient of the image, as detailed in Section 3.2.2, and an implicit level set curve evolution.

When t tends to $+\infty$, γ_t converges to a curve that is a local minimizer of the weighted length L. One should be careful, and note that this curve is not a globally minimal path for the metric, as defined in (1.4). Indeed, for the case of a closed curve, a globally minimal closed curve would be restricted to a single point. To avoid the curve to shrink toward itself, one can add an artificial velocity that inflates the curve [76], called the pressure force or balloon force.

3.1.2 Minimal Paths

A major difficulty with the active contour approach is that the curve γ_t evolving in time might be trapped in poor local minima of the energy

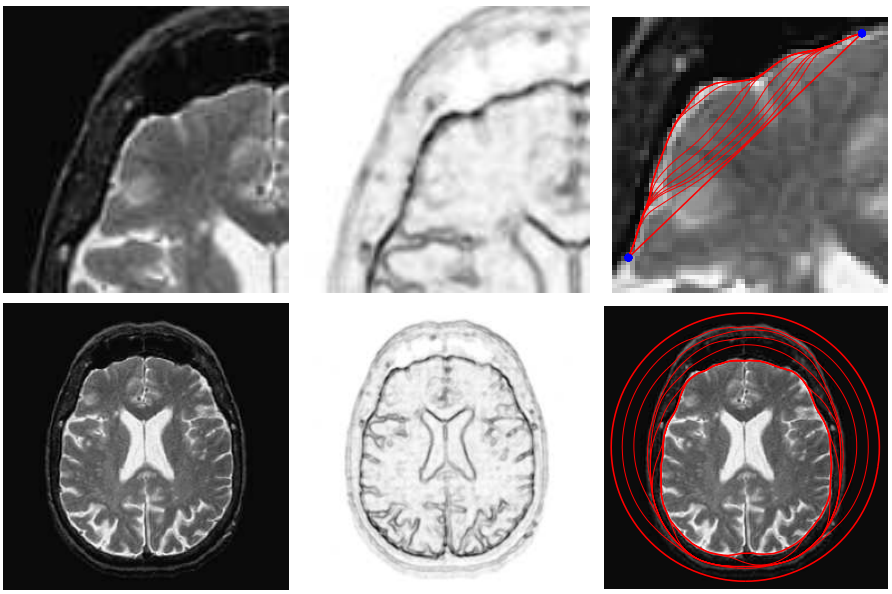


Fig. 3.4 *Example of geodesic active contour evolution for medical image segmentation. Top row: open contour, bottom row: closed contour. The potential W is computed using* (3.12) *with* $\alpha = 1$ *and* $\varepsilon = 10^{-3}\|\nabla f\|_\infty$.

E, thus leading to a bad segmentation. It is especially the case for noisy images such as in medical imaging applications.

In the case of an open curve, subject to the boundary conditions (3.2), Cohen and Kimmel [70] use the Fast Marching propagation to find the global minimum of the energy $E(\gamma) = L(\gamma)$. Boundary constraints forbid the segmentation of closed object with a single curve, but allow to track curvilinear features such as roads in satellite imaging or vessels in medical imaging. Notice that [70] also proposed a way to find a closed curve as the union of two geodesics. This was followed by other approaches to define a closed curve as a set of geodesics [77, 24]. The keypoint approach of [24] allows to give only a starting point on the boundary of an object and find the complete closed contour, see Figures 3.6 and 3.7.

The curve $\gamma \in \mathcal{P}(x_s, x_e)$ minimizing $L(\gamma)$ is the geodesic minimal path $\gamma^\star$ already defined in (1.4). It can be computed as detailed in Sec-

tion 1.5 by computing the distance map U_{x_s} and performing the gradient descent (1.26). The distance map U_{x_s} is computed in $O(N^2 \log(N))$ operations for an image of N^2 pixels using the Fast Marching algorithm detailed in Section 2.3.

Figure 3.5 shows an example of the extraction of a tree of vessels, that are shortest paths joining several end points to a single starting point. The metric $W(x)$ is computed by applying some standard image processing techniques to f. In this case, the background is subtracted by applying a high-pass filter, and the filtered image is thresholded to increase the contrast of the vessels. The following section details various methods to compute a metric W adapted to an image to analyze.

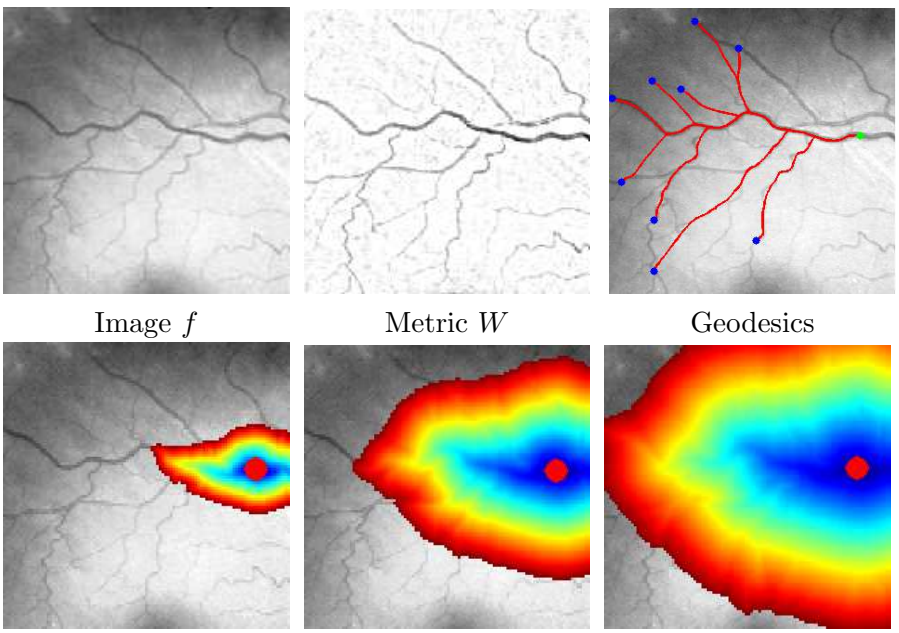


Fig. 3.5 *Example of minimal path for vessel extraction. The bottom row shows the evolution of the Fast Marching propagation.*

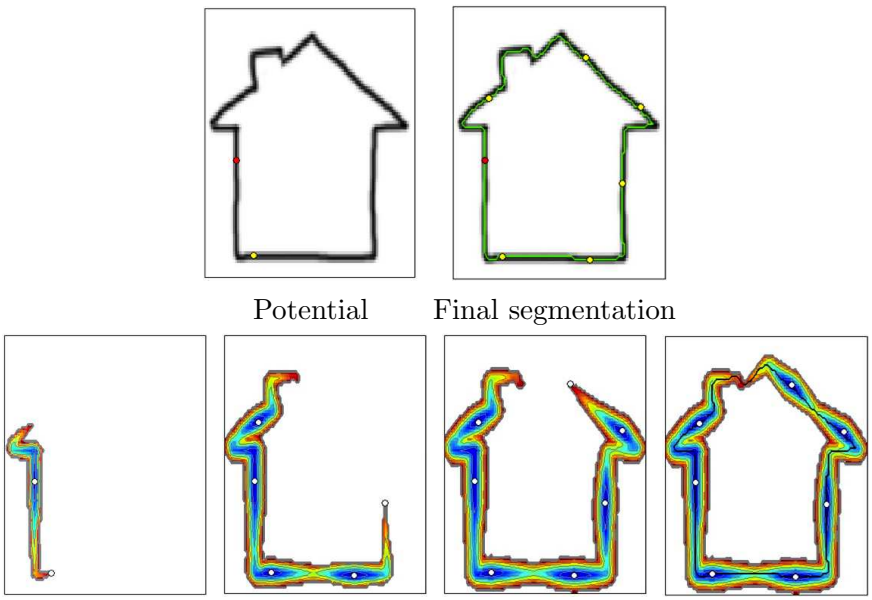


Fig. 3.6 *Illustration of the method of minimal paths with keypoints of [24]. A start point is given (top left image, red point) and a set of keypoints (top right image, yellow points) is obtained automatically to segment the closed boundary of an object. The key points are seeded by using iteratively front propagations as shown on the bottom row.*

3.2 Metric Design

In practice, the difficult task is to design a metric W in order to have meaningful geodesics. Here are some examples of possible choices, for the processing of an input image f.

3.2.1 Intensity-based Metric

Starting from an image $f : [0,1]^2 \to \mathbb{R}$, the basic idea is to compute roads or vessels as shortest paths in the plane of the image. A potential must be designed such that computed shortest paths correspond to actual roads or vessels in the images.

A natural idea is to design the potential depending on the value of the image

$$W(x) = W_0 + \rho(f(x)), \tag{3.10}$$

where $\rho : \mathbb{R} \mapsto \mathbb{R}^+$, $\min_a \rho(a) = 0$. The constant $W_0 > 0$ is regularizing the geodesic curve by penalizing their Euclidean length.

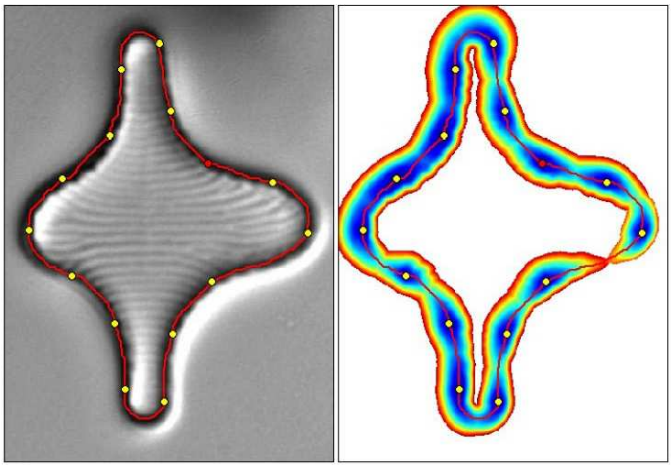


Fig. 3.7 *Illustration of the method of minimal paths with keypoints of [24]. A start point is given in red and a set of keypoints is obtained automatically to segment the closed boundary in this biology image.*

Since in most medical images, vessels can be made darker than the background, ρ should be a non-decreasing value of the image intensity. Doing so, shortest paths are likely to follow dark areas of the images, *i.e.* vessels. This is illustrated in fig 3.8.

In other applications, such as satellite images, the curves of interest are assumed to be of approximately constant gray value c. In this case, one can choose for instance

$$W(x) = W_0 + \rho(f(x)) \quad \text{with} \quad \rho(a) = |a - c|^\alpha \tag{3.11}$$

where α is tuned depending on the characteristics of the image and on the confidence one has about c. Figure 1.1 shows an example of road extraction, where the metric (3.11) is used with $\alpha = 1$.

3.2.2 Gradient-based Metric

In several applications, the curves of interest are located near areas of large variation of intensity in the image – e.g. when one wants to detect the boundary of object in images. In this case, one can choose a gradient based potential, such as for instance

$$W(x) = \rho(\|\nabla f(x)\|) \quad \text{where} \quad \nabla f(x) = \left(\frac{\partial f}{\partial x_1}, \frac{\partial f}{\partial x_2} \right) \in \mathbb{R}^2, \tag{3.12}$$

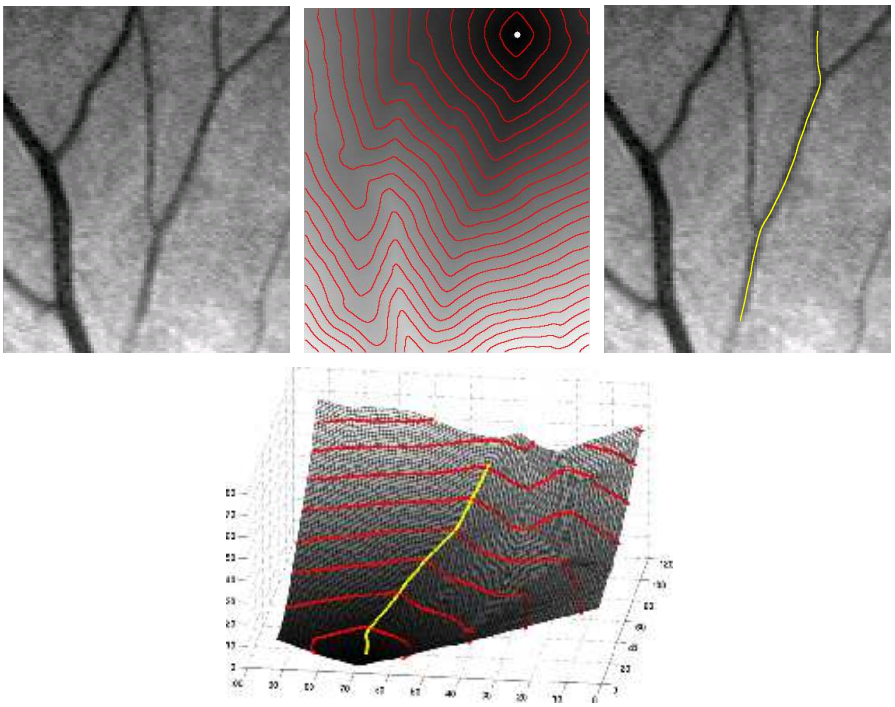


Fig. 3.8 *Vessel segmentation using shortest paths. Top left: original retinal image from DRIVE database [199]. Top middle: distance map computed from the white point (gray level was used as potential) and iso-distance lines (red). Notice that the front propagates faster along the vessels. Top right: shortest path computed from another point of the vessel. Bottom: synthesis on the distance function elevation map.*

where ρ is a non increasing function such as

$$\rho(a) = (\varepsilon + a)^{-\alpha} \tag{3.13}$$

for some contrast parameter $\alpha > 0$. This corresponds to an edge attracting potential. The gradient vector ∇f is estimated numerically using finite differences, possibly after smoothing the original image to remove some noise.

Figure 3.9 shows an example of use of the gradient based metric (3.13) with $\alpha = 1$. A set of two initial points S linked by four geodesics to two other points, to obtained a segmentation of the cortex with a closed curve.

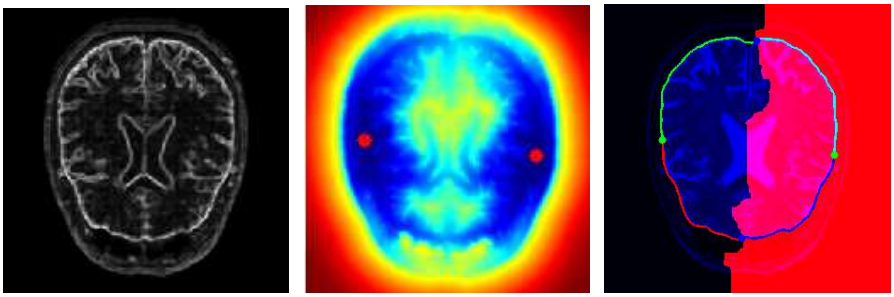


Fig. 3.9 *Example of segmentation using minimal paths for a gradient-based metric.*

3.2.3 Volumetric Metric

The Fast Marching works the same way in any dimension and in particular can be used to extract shortest paths in 3D volumetric medical data [94]. Such a volume is a discretization of a mapping $f : [0,1]^3 \mapsto \mathbb{R}$.

Figure 3.10 shows some examples of geodesic extraction on a medical image that represents tubular structures (blood vessels) around the heart. Since a pixel x inside a vessel has approximately a known intensity value $f(x) \approx c$, the potential $W(x)$ is defined as in (3.10).

Different extensions of minimal paths have been proposed in [11, 12] in order to find a surface between two curves in a 3D image. These approaches are based on defining a network of minimal paths between the two curves, see Figure 3.11 A transport equation was used to find this network efficiently without computing the paths themselves.

3.2.4 Metrics on 3D Surfaces

Salient features on a surface $\mathcal{S} \subset \mathbb{R}^3$ can be detected by extracting geodesics that are constrained to be on the surface. As detailed in Section 1.2.1, for a parameterized surface $\varphi : \Omega \to \mathbb{R}^3$, this corresponds to an anisotropic Riemannian metric on the parametric domain Ω, whose tensor is the first fundamental form (1.1.2).

As in Section 1.2.1, one should be careful about the distinction between a point $x \in \Omega \subset \mathbb{R}^2$ in the parameter domain, and its mapping $\tilde{x} = \varphi(x) \in \mathcal{S} \subset \mathbb{R}^3$ on the surface.

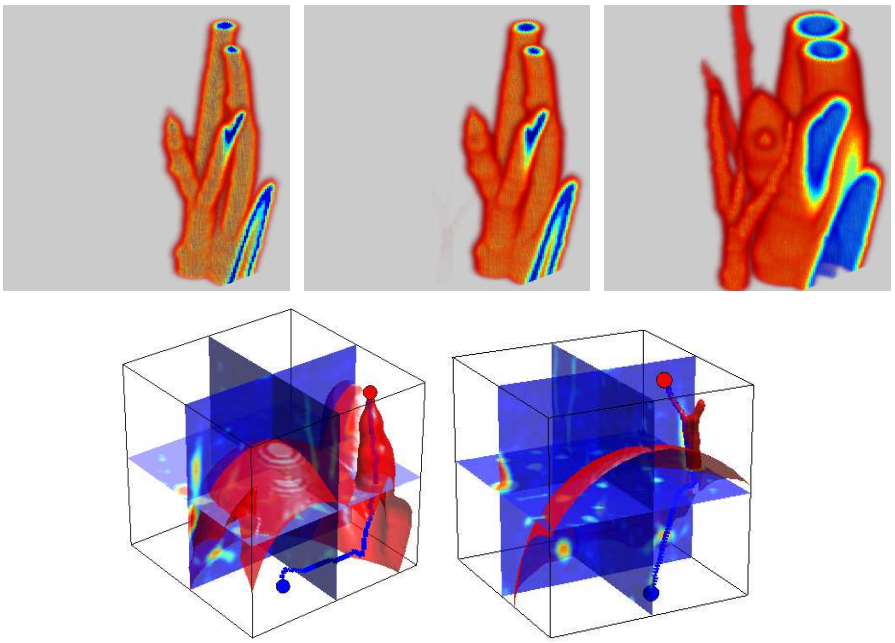


Fig. 3.10 *Example of volumetric Fast Marching evolution (top row) and geodesic extractions (bottom row).*

Metric on textured meshes. In order for the geodesic to follow salient features, the length $L(\tilde{\gamma})$ of a curve $\tilde{\gamma} \in \mathcal{S}$ is measured as in (1.5) using a weight $W(\tilde{x}) > 0$ for each point $\tilde{x} \in \mathcal{S}$. The simplest way to define this metric is using some texture function $f(\tilde{x}) \in \mathbb{R}$ defined on the surface. Following (3.10) or (3.12), one can define metrics based either on the gray levels or on the gradient of f.

Figure 3.12 shows the influence of a varying metric $W(\tilde{x})$ on the geodesic that follows the salient features of the texture.

Curvature-based metric. To detect geometric salient features on surfaces, such as ridges and valleys, one needs to use higher order derivatives of the surface parameterization, see for instance [204]. One typically uses curvature information, that corresponds to extrinsic quantities that depend on the embedding of the surface in $\mathbb{R}^3$.

Curvatures are computed by measuring the second order variation

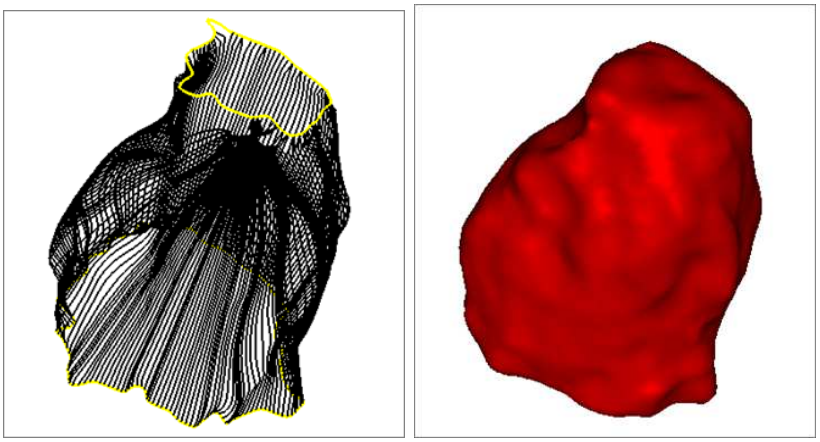


Fig. 3.11 *Illustration of the method of minimal paths network [11] in order to find a surface as a set of 3D minimal paths between two given curves. On the left, the two curves and network are shown. On the right the surface obtained by interpolation from the path network.*

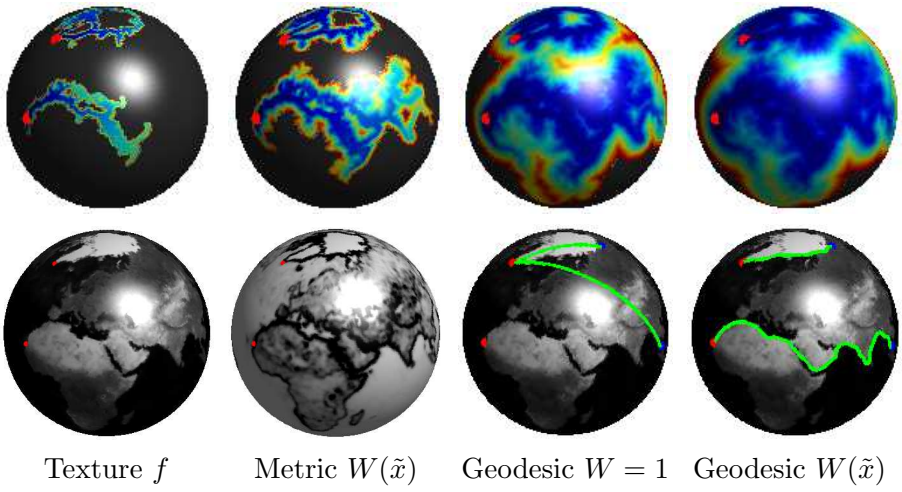


Fig. 3.12 *Top row: Fast Marching propagation from the two red points for a texture-based metric $W(\tilde{x})$. Bottom row: comparison of geodesic curves from the two blue points for the constant metric $W = 1$ and for a texture-based metric $W(\tilde{x})$.*

of the surface projected on the unit normal to the surface

$$\forall\, x \in \Omega, \quad n(x) = \frac{\bar{n}(x)}{\|\bar{n}(x)\|} \quad \text{where} \quad \bar{n}(x) = \frac{\partial \varphi}{\partial x_1}(x) \wedge \frac{\partial \varphi}{\partial x_2}(x)$$

where $\wedge$ denotes the cross product in $\mathbb{R}^3$. The second fundamental form is then defined as

$$\forall\, x \in \Omega, \quad J_\varphi(x) = \left\{ \langle \frac{\partial^2 \varphi}{\partial x_i \partial x_j}(x),\, n(x) \rangle \right\}_{1 \leqslant i,j \leqslant 2} \in \mathbb{R}^{2\times 2}. \tag{3.14}$$

If the parametrized surface is smooth enough, this second fundamental form is a symmetric matrix, that can be diagonalized as

$$J_\varphi(x) = \mu_1(x) e_1(x) e_1(x)^{\mathrm{T}} + \mu_2(x) e_2(x) e_2(x)^{\mathrm{T}}. \tag{3.15}$$

The eigenvectors $e_i(x) \in \mathbb{R}^2$ for $i = 1, 2$ are orthogonal, and the tangent vectors $D\varphi(x)e_i(x)$ are the principal curvature directions of curvature on the surface, where $D\varphi(x)$ is the differential of the parameterization defined in (1.8).

This factorization (3.15) should not be confused with the decomposition (1.16) of the Riemannian tensor T_x computed from the first fundamental form $I_\varphi(x)$. In particular the eigenvalues μ_i can be negative.

The second fundamental form $J_\varphi(x)$ can be approximated numerically on 3D triangulated meshes using averaging of rank-1 tensors [78] or using a local covariance analysis [66, 229].

A weighting function $W(\tilde{x})$ can be defined from the eigenvalues $(\mu_1(x), \mu_2(x))$ so that the geodesics follow local extrema of some curvature measure. For instance, one can use

$$W(\tilde{x}) = \rho(\mu_1(x)^2 + \mu_2(x)^2) \tag{3.16}$$

where ρ is a non-increasing function, such as (3.13). Figure 3.13 shows the computation of the distance map to a set of starting points using the Fast Marching propagation. The front moves faster along features when W takes into account the curvature. Figure 3.14 shows how this curvature-based metric influences the minimal paths, that are forced to follow the salient features of the surface.

3.2.5 Anisotropic Metrics for Images and Volumetric Datasets

In order to better follow the salient structures of an image or a volume f, one can replace the isotropic metric $T_x = W(x)^2 \mathrm{Id}_d$ by an anisotropic metric $T_x \in \mathbb{R}^{d\times d}$.

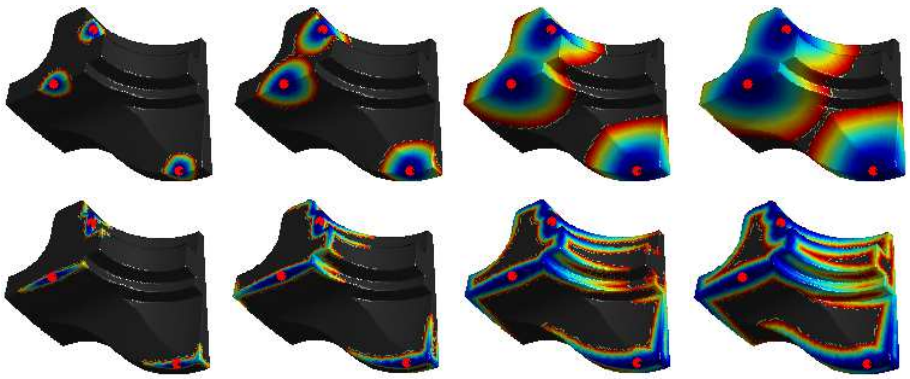


Fig. 3.13 *Top row: Fast Marching propagation from the red points for the metric* $W = 1$. *Bottom row: propagation for the metric* $W(\tilde{x})$ *defined in* (3.16).

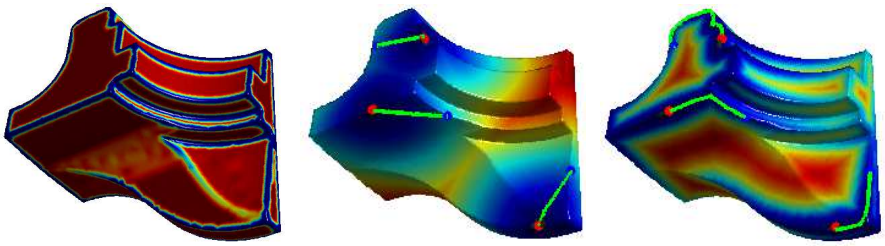


Fig. 3.14 *Comparison of the geodesics for* $W = 1$ *and for the curvature based metric* $W(\tilde{x})$ *defined in* (3.16).

In some applications, a hardware acquisition device actually gives access to a local anisotropic metric. This is for instance the case in medical imaging for Diffusion MRI (dMRI)[17], in which case $d = 3$.

This modality derives from MRI and aims at computing the probability distribution of water molecules diffusion at any point of a human brain over the set $\mathcal{S}^2$ of all possible directions in 3D. Since water molecules tend to propagate faster along white matter fibers, dMRI allows to obtain a local map of white matter fibers directions.

The most commonly used model for the probability distribution of water diffusion is the diffusion tensor (DTI) [16], which simply consists in a 3-dimensional tensor. While this rough representation does not allow to recover precise information when fibers crossings or splittings

occur, it performs well when only one bundle of fibers is present at a precise location. Its main advantage is that only scans in the 3 principal directions are needed to recover a diffusion tensor, which results in a low data acquisition time.

After data acquisition and computation of the diffusion tensor, every point x in the white matter is thus equipped with a tensor $\mathcal{D}_x$ whose principal eigenvector gives an evaluation of the main direction of fibers at this point (Figure 3.15).

Extracting full length fibers numerically from this local information is important to compute a map of the white matter and detect pathologies.

Full length fibers can be modeled as geodesic curves inside the white matter for the Riemannian metric $T_x = \mathcal{D}_x^{-1}$: geodesics in such a Riemannian space will tend to align themselves with the eigenvector corresponding with the smallest eigenvalues of T_x [231, 215, 139]. They will thus follow high-diffusion paths, which are likely to correspond to white matter fibers (Figure 3.16).

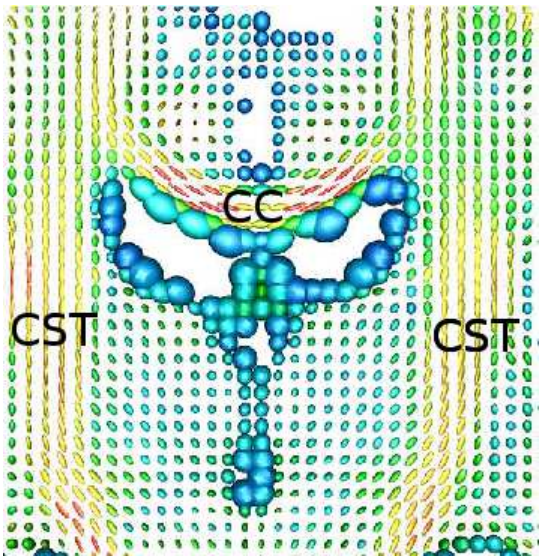


Fig. 3.15 *DTI of a human brain on a coronal slice. Corpus Callosum (CC), which fibers have an horizontal orientation, and the left and right Corticospinal Tract (CST), which fibers have a mainly vertical orientation, can be seen in the plane of the image.*

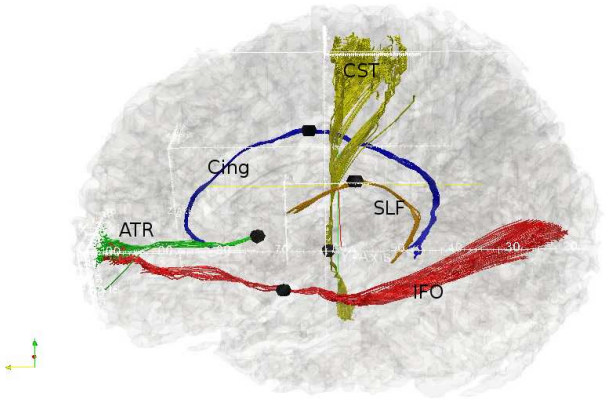


Fig. 3.16 *Geodesics corresponding to major white matter fibers bundles in left hemisphere, obtained with the geodesic method of [215].*

3.3 Centerlines Extraction in Tubular Structures

In many applications, such as road tracking or vessel extraction, one is interested in computing the centerline of the tubular structures of interest. As illustrated in Figure 3.8 and 3.17, minimal paths tend to follow the boundary of the vessel in regions where the tubular structure is curved. Furthermore, one is often interested in computing an estimation of the radius of the vessels, which evaluation may have medical significance, *e.g.* in retinal imaging [299] or for detecting stenosis. The precise value of this radius is not directly accessible using minimal paths.

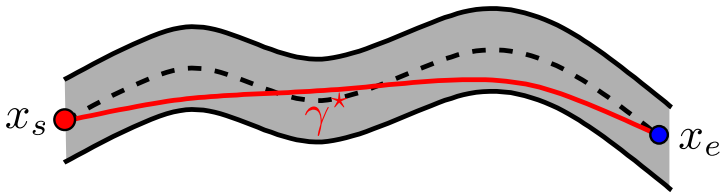


Fig. 3.17 *The path centering problem. The centerline is displayed as a dashed curve.*

3.3.1 Centering the Geodesics

A simple way to re-center the minimal path is to make the potential locally minimal in the middle of the tubular structure. For symmetric vessels, a possibility is to smooth the potential, and replace $W(x)$ defined in (3.10) by

$$W(x) = \varepsilon + (\rho(f) \star G_\sigma)(x), \tag{3.17}$$

where G_σ is a Gaussian kernel of width $\sigma > 0$. This smoothing is however difficult to control, and leads to a loss in the resolution of the image.

Another option is to re-center the geodesic in a post-processing step. Deschamps and Cohen [94] perform this re-centering using a two step method. Starting from the initial non-centered geodesic, the boundary $\partial\mathcal{V}$ of the vessel $\mathcal{V}$ is extracted using the front of the Fast Marching. A modified potential is then defined as

$$W(x) = \begin{cases} \varepsilon + \rho(d(\partial\mathcal{V}, x)), & \text{if} \quad x \in \mathcal{V} \\ +\infty \quad \text{otherwise} \end{cases},$$

where $d(\partial\mathcal{V}, x)$ is the geodesic distance to the boundary of the vessel, and ρ is a non-increasing mapping. This new potential forces minimal paths to follow the center of the tubular structure. This method has been applied to enable virtual endoscopy exploration of vessels in [94, 75], see Figures 3.18 to 3.20.

3.3.2 High Dimensional Lifting

To automatically center the minimal paths and compute the local radius of the vessel, one can compute a metric W on a higher dimensional space.

Radial lifting. Li et al. [168] proposed extracting minimal paths

$$\gamma^\star = (\gamma_x^\star, \gamma_r^\star) \in \Omega \times [r_{\min}, r_{\max}]$$

on a space with an additional radius dimension. See also [227] for a similar approach using geodesic computation on graphs.

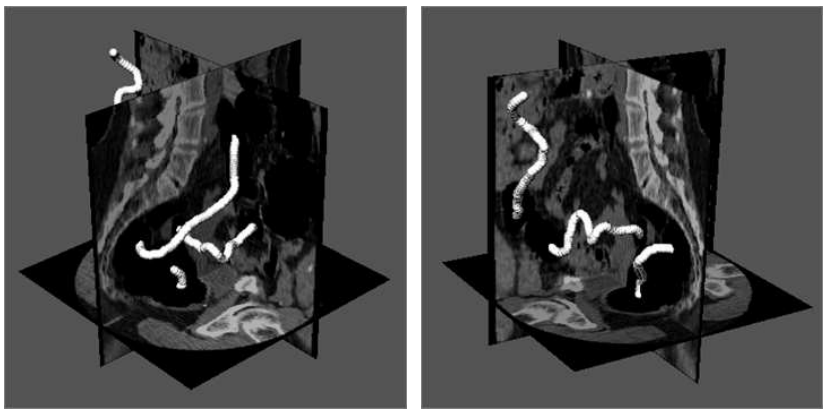


Fig. 3.18 *Example of a centered path inside the colon in a 3D image. Two different views of the path together with slices of the image.*

The geodesic curves are defined with respect to an isotropic Riemannian metric $W(x,r) > 0$ for $x \in \Omega$ and $r \in [r_{\min}, r_{\max}]$. The spatial geodesic $\gamma_x^\star$ is intended to follow the centerline of the vessel, and $\gamma_r^\star(t)$ should indicate the local radius of the vessel near $\gamma_x^\star(t)$. This is achieved if $W(x,r)$ is locally minimal at a point x if a vessel is centered at x and the radius of the vessel is approximately r.

A local detector $D(x,r)$ evaluates the likelihood of the presence of the centerline of a vessel of radius r at every point x of the image. The value of $D(x,r)$ is computed in [168] by measuring the deviation of the mean and variance of the gray value inside a sphere centered at x and of radius r. The metric is then defined as $W(x,r) = \rho(D(x,r))$ where ρ is a decreasing function.

Because a spherical region is a poor match with the anisotropic geometry of vessel, this method is sensitive to initialization and parameters. Figure 3.21 shows an example of result of the method in [168] together with its extension in [169] using the keypoints of [24]

Radial and orientation lifting. Pechaud et al. proposed in [216, 214] to lift a 2D image to a 4D space that includes both a radius information $r \in [r_{\min}, r_{\max}]$ and an orientation $\theta \in [0, \pi)$.

The method makes use of a basic template $M(x)$ that is scaled and

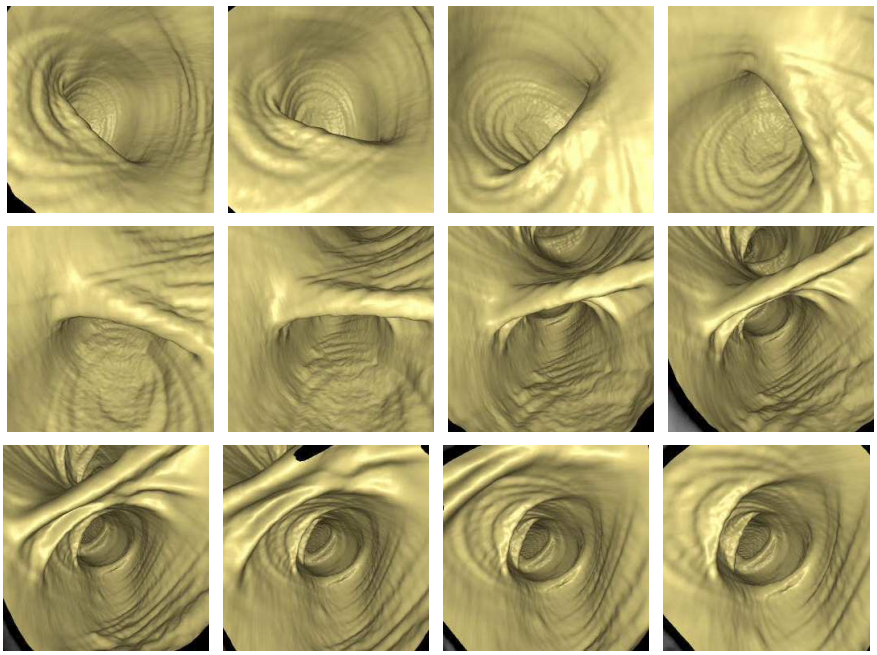


Fig. 3.19 *Some images of a virtual flythrough along the 3D minimal path of the previous figure. Each view is obtained as what a virtual camera would see on a given point of the path in the direction tangential to the path. The only inputs are the 3D image, a starting point and a given length of the path.*

rotated

$$\forall\, x \in \Lambda(r,\theta), \quad M_{r,\theta}(x) = M(R_{-\theta}(x)/r),$$

where $\Lambda(r,\theta)$ is the scaled and rotated domain over which the template is defined, see Figure 3.22, left. This basic element M is specific to a given application, and might be different if one considers roads (that are usually bright on a dark background, with sharp edges) or vessels (that may have a reversed contrast and blurred edges).

The local detector $R(x,r,\theta)$ is computed as a cross-correlation

$$R(x,r,\theta) = \xi_{\Lambda(r,\theta)}(M_{r,\theta}(\cdot), f(x+\cdot))$$

where $f(x+\cdot)$ is the image translated by x, $\xi_A(f,g)$ is the normalized cross-correlation between f and g over the domain A, defined by:

$$\xi_A(f,g) \stackrel{\text{def.}}{=} \frac{\int_A (f-\bar{f})(g-\bar{g})}{\sqrt{\int_A (f-\bar{f})^2}\sqrt{\int_A (g-\bar{g})^2}} \tag{3.18}$$

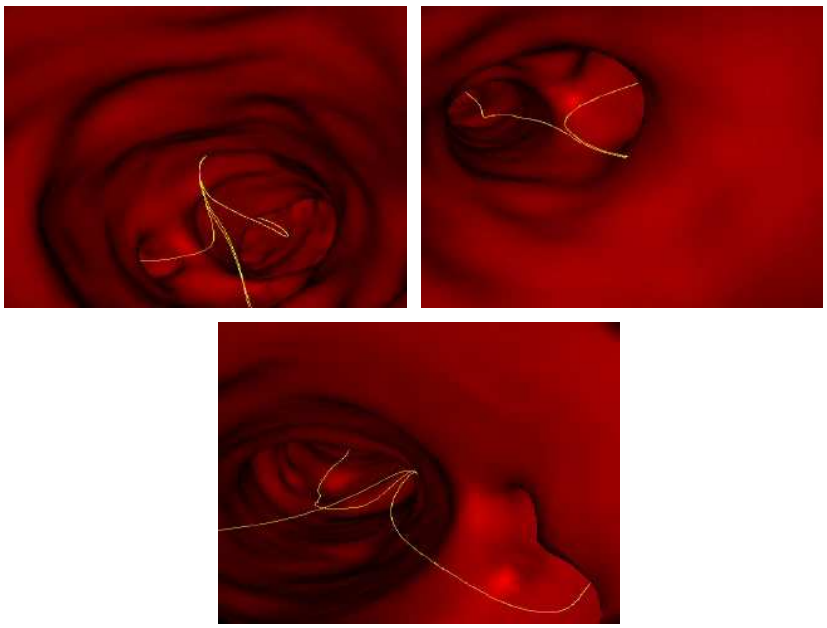


Fig. 3.20 *Virtual endoscopy through centered paths inside the aorta tree structure in a 3D image. Different paths are shown in yellow that link the source point to different endpoints of the structure.*

where $\bar{h} = \frac{\int_A h}{\mu(A)}$, $\mu(A) = \int_A 1$ being the area of A.

Figure 3.22 shows an example of this 4D lifting. The additional angular dimension helps to disambiguate situations where vessels with different orientations are crossing.

The metric is then defined as $W(x, r, \theta) = \rho(R(x, r, \theta)) > 0$, where ρ is a decreasing function. Geodesics are then extracted in this lifted space

$$\gamma^\star = (\gamma_x^\star, \gamma_r^\star, \gamma_\theta^\star) \in \Omega \times [r_{\min}, r_{\max}] \times [0, \pi)$$

where the angular dimension $[0, \pi)$ should be handled with periodic boundary conditions. Domain $\Omega \times [r_{\min}, r_{\max}] \times [0, \pi)$ can thus be seen as a 4D cylinder, which could be embedded in a 4D Euclidean space.

Figure 3.23 shows centerlines $\gamma_x^\star$ and radii obtained with this method.

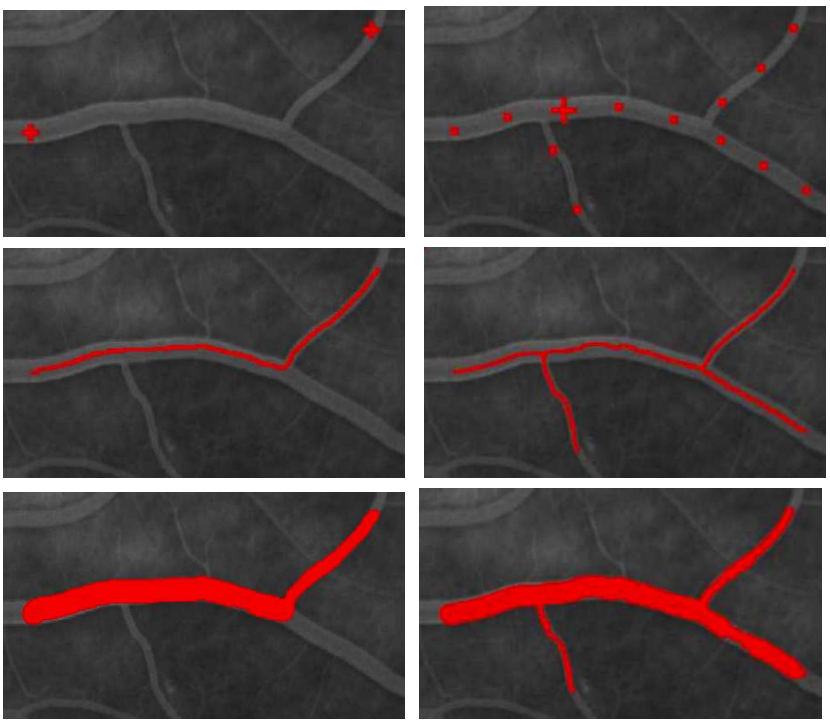


Fig. 3.21 *Examples of minimal paths in the 2D+radius space for segmentation of vessels and their centerlines. On the left two endpoints are given and we obtain a centerline together with a region. On the right only one starting point is given and a set of keypoints is obtained automatically to segment the vascular tree structure.*

Anisotropic lifting. Benmansour and Cohen proposed in [23] to reduce the numerical complexity of centerline extraction by building an anisotropic Riemannian metric $T_{x,r}$ for each space and scale location $(x, r) \in \Omega \times [r_{\min}, r_{\max}]$.

The orientation selectivity of the metric replaces the orientation dimension $\theta \in [0, \pi)$ in (3.3.2). This can be achieved by computing for each (x, r), a metric $T_{x,r} \in \mathbb{R}^{2\times 2}$ such that, for all unit vector $n_\theta = (\cos(\theta), \sin(\theta))$, the anisotropic potential $\|n_\theta\|_{T_{x,r}}$ is close to a criterion $W(x, r, \theta)$ similar to the one above. An alternative construction based on multi-scale second order derivatives of Gaussian filtering was proposed in [23] to efficiently compute the metric $T_{x,r}$. The response of the filter is designed to be maximal on the centerline of vessels, and the tensor principal eigenvector corresponds to the estimated direction

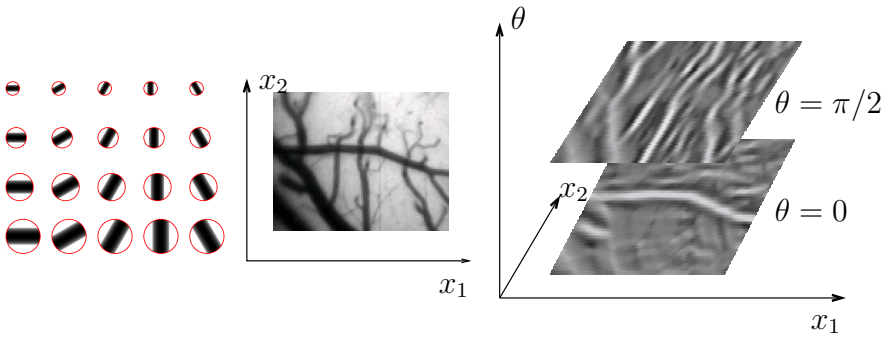


Fig. 3.22 *Left: vessel template at different scales and orientations. Middle: original 2D image. Right: 4D lifting $R(x_s, r, \theta)$ (fixed radius r), ranging from -1 (black) to 1 (white).*

of the vessel, see Figures 3.24 and 3.25.

3.4 Image Segmentation Using Geodesic Distances

3.4.1 Segmentation Using Geodesic Balls

Active contour by geodesic ball growing. In the active contour model (3.3), the curve evolution (3.7) is allowed to move with positive or negative speed. Furthermore, this speed β depends both on the position of the curve, and on its orientation. If one imposes a simpler evolution model, where the evolution is performed with a strictly positive speed $W(x) > 0$

$$\frac{d}{dt}\gamma_t(s) = W(\gamma_t(s))n_t(s), \tag{3.19}$$

then this evolution can be tracked using level sets of a single distance function U_S,

$$\{\gamma_t(s) \setminus s \in [0,1]\} = B_t(x_s) = \{x \in \Omega \setminus U_S(x) = t\}, \tag{3.20}$$

where the initial curve γ_0 at $t = 0$ is the boundary of the starting points

$$\partial S = \{\gamma_0(s) \setminus s \in [0,1]\}$$

and where U_S is the geodesic distance map to S for the isotropic metric $W(x)$, as defined in (1.22).

The curve γ_t is thus the boundary of a geodesic ball of radius t. It can thus be computed using the Fast Marching, and in fact, γ_t can be

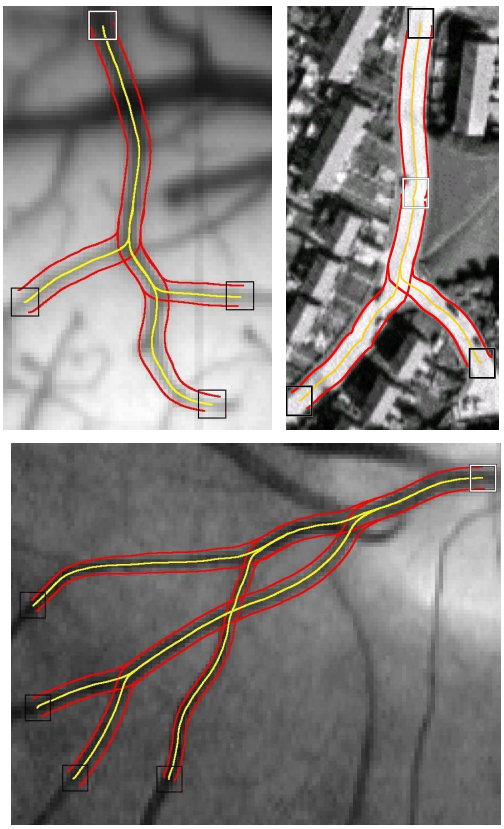


Fig. 3.23 *Centerlines positions and radii extraction of vessels in a cortical image (top left), in a satellite (top right) and in a retinal image (bottom). White square denotes the initial points* $S = \{x_s\}$*, while black squares are different ending points.*

approximated by the front that the Fast Marching propagates during the iterations.

As t increases, this ball γ_t inflates, and moves faster in region where W is large. Malladi and Sethian [178] thus propose to use this evolution to segment object in noisy images, using a metric $W(x)$ that is low for pixel x outside the object to detect, and using a radius t chosen to match the size of the object. See also [94, 181] for applications of this method to medical imaging.

Figure 3.26 shows application of this method to segment a medical image f using a pixel-based potential (3.11), and where the initialization is performed with a single point $S = \{x_s\}$ located inside the region

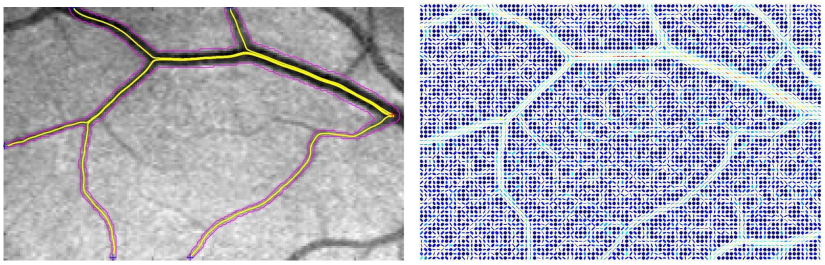


Fig. 3.24 *Examples of minimal paths in the 2D+radius space for segmentation of vessels and their centerlines. The metric based on the Optimal Oriented Flux is shown on the right.*

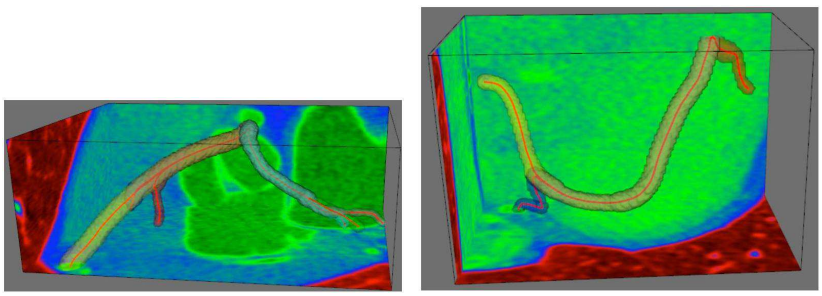


Fig. 3.25 *Examples of minimal paths in the 3D+radius space for segmentation of vessels and their centerlines.*

to segment.

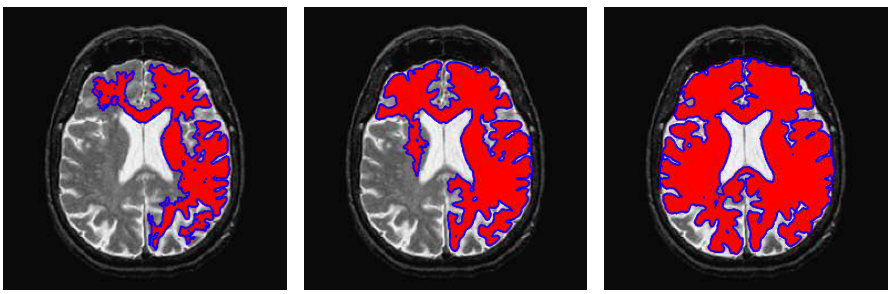


Fig. 3.26 *Front propagation of the Fast Marching algorithm, the region indicates the geodesic ball* $\{x \in \Omega \setminus U_S(x) \leqslant s\}$*, for* s *increasing from left to right. The boundary of this region corresponds to the curve* γ_s*.*

Front freezing. Geodesic balls have the tendency to extend beyond the boundary of the object. This is especially the case for elongated tubular structure such as vessels. To avoid this leaking artifact of the front during the propagation, Deschamps and Cohen [95, 75] proposed to freeze the propagation in regions where the front is moving slowly. This modification improves the quality of the segmentation, although the segmented region is not defined as a geodesic ball anymore.

3.4.2 Segmentation Using Geodesic Voronoi Regions

To segment several regions, an approach consists in giving a set of seed points $\{x_i\}_i$, where each x_i is supposed to be inside a region. One then considers the segmentation of the image domain into Voronoi cells, as defined in (1.18). As explained in Section 2.6.1, this segmentation can be computed approximately during the Fast Marching propagation, or during iterative schemes. See for instance [262, 172, 200, 10] for an application of this method to perform image segmentation.

For this segmentation to be efficient, the front should move slowly for pixels that are intended to be at the boundary between several regions. For object separated by edge transition, this can be achieved by using an edge stopping metric

$$W(x) = \rho(\|\nabla f(x)\|),$$

where ρ is an increasing function. Note that this potential is inverse to the edge attracting criterion defined in (3.12) .

Figure 3.27 shows an example of segmentation using a contrast function $\rho(a) = (\varepsilon + a)^\alpha$, for $\alpha = 1$ and a small $\varepsilon > 0$.

Relation with watershed. If the geodesic distance map U_S is approximated with morphological operators, as detailed in Section 2.8.1, the Voronoi segmentation corresponds to the result of the watershed algorithm initially proposed in [28] and extended for instance in [283, 198, 188]. For the watershed algorithm, the set of initial points S is usually chosen as the local minima of the map $W(x)$, possibly after some smoothing pre-processing.

The Fast Marching implementation of the Voronoi segmentation tends to perform more precise segmentations since it does not suffer

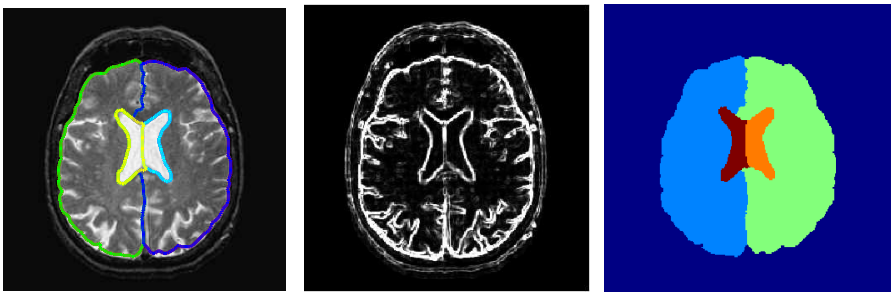


Image f Metric $W(x)$ Segmentation

Fig. 3.27 *Image segmentation using Voronoi regions. The colored curves on the left image show the boundary of the Voronoi cells.*

from metrication caused by the discrete nature of the structured element of mathematical morphology operators, see [189].

3.5 Shape Offsetting

Shape offsetting is an important issue, both in motion-planning and CAD [241]. Starting from a set $A \subset \Omega \subset [0,1]^2$ or $[0,1]^3$ and a ball B of radius r centered at 0, it consists in finding the set $\delta_B A = \{x + b \mid x \in A, b \in B\}$ described in Section 2.8.1 (see Figure 3.28).

Fig. 3.28 *Illustration of shape offsetting. Starting from the black set A and the red ball B (left), $\delta_B A$ is constructed (middle and right.)*

When the boundary of A consists in segments or circular arcs, exact methods have been developed in order to compute its offsetting [241]. In the general case however, such methods are not available, and one must rely on more general algorithms. Early methods are based on level sets [149].

As explained in this section, this operation can be approximately performed using mathematical morphology in $O(n)$ time if Ω is convex.

If Ω is not convex, one can approximate the shape offsetting by using the geodesic dilation of Section 2.8.1. However, a more precise alternative consists in using the methods described in Section 2.2. The starting set S is A. One thus has $\delta_B A = \{x \in \Omega, U_S(x) \leqslant r\}$. This set can be easily computed as the set of *Computed* points of the Fast-Marching algorithm when the first point with distance greater than r has been reached [151].

Notice that this easily extends to the case when B is elliptic. One needs to apply an affinity f to B such that $f(B)$ is a sphere. It is then possible to prove that $\delta_B(A) = f^{-1}(\delta_{f(B)}(f(A)))$. In order to compute $\delta_B(A)$, one thus deforms the space A through f, computes its offsetting as explained before, and brings back the result in the initial space through the application f^{-1}.

3.6 Motion Planning

Computation of shortest paths inside a shape with Euclidean metric has been thoroughly studied in motion planning [161]. Solving this problem indeed allows to move a robot from one point to another in an environment with obstacles, see Figure 3.29, left.

The methods detailed in Chapter 2 are rarely used in motion planning, mainly for three reasons : firstly, one is interested in finding some path between two points, not necessarily the shortest one. Secondly, in robot-motion planning, the environment is often discovered progressively while the robot moves, which makes it impossible to use a geodesic method. Thirdly, since the space is Euclidean, taking into account the possible shapes of the obstacles allows to design ad-hoc methods to compute shortest paths. However, some examples exist where the Fast-Marching algorithm is used in the context of motion planning – e.g. to model crowd movements [273].

Punctual object As an example, assume that one wants to compute the optimal trajectory of a punctual object in a polygonal environment Ω (see Figure 3.29, left.) This corresponds to a specific case of the problem described in Section 1.3.2. An efficient method exists to compute an exact solution : one can compute the so-called shortest-path road-

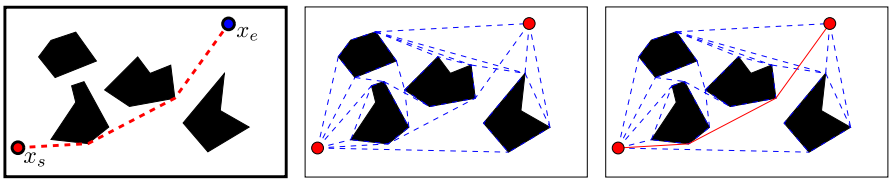


Fig. 3.29 *Shortest path in a space with polygonal obstacles. A polygonal environment with initial and ending points (left), the shortest-path-roadmap (middle), and the shortest path between initial and ending points (right).*

map G [160], which is actually a peculiar graph linking the obstacles vertices and the initial and ending points (see Figure 3.29, middle). It is possible to show that the shortest path between the initial and ending points consists of edges of G (see Figure 3.29, right). Computing the geodesic between two points thus boils down to computing a path in the adjacency graph of the cells. The whole operation can lead to the exact geodesic in $\mathrm{O}(n^2 \ln n)$ time, where n stands for the number of edges in the environment.

Spherical object Shape offsetting (see Section 3.5) can be used to compute trajectories of non-punctual objects in an environment with obstacles. Assume that a circular object B_r of radius r is to be moved in an environment with obstacles (Figure 3.30, top). The problem is thus to find a trajectory γ of the center of B_r such that $\delta_{B_r}(\gamma) \subset \Omega$, where δ_{B_r} corresponds to the dilation of the curve γ defined in Section 2.8.1.

This problem can be reduced to the previous one by growing the obstacles by a distance r. One defines $\Omega' = \delta_{B_r}(\Omega)$ (Figure 3.30, middle.)

Computing B_x trajectory in Ω is clearly equivalent to computing the trajectory of a single point in Ω' (Figure 3.30, right.)

However, geodesic methods can be used to perform the offsetting only when the object to move is circular or elliptic, which limits its practical utility.

3.7 Shape From Shading

Shape from shading is a popular computer vision inverse problem, see for instance [297] for a review on this subject. It corresponds to the

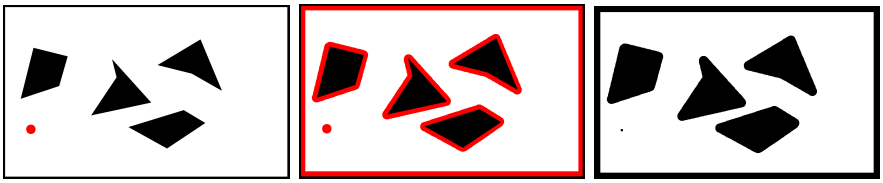


Fig. 3.30 *Illustration of shape offsetting. Assuming one wants to compute the trajectory of a circular object in an environment Ω with obstacles (top), one can grow the obstacles, leading to a new environment Ω' (middle), and the problem is thus reduced to computing the trajectory of a point in the obtained space (right.)*

process of recovering a 3D surface from a shaded image of that surface. In simplified settings, the surface is obtained by solving a non-linear PDE, as first noticed by Horn [135]. Although it is not related to the extraction of minimal path, in some cases, it can be cast in an Eikonal equation, and can thus be solved using Fast Marching methods.

In a simple setting, one wants to recover a surface, represented as height field $u(x) \in \mathbb{R}$ for $x = (x_1, x_2) \in [0,1]^2$ from a 2D picture $f(x)$. A simplifying assumption is that the surface is Lambertian diffuse. Ignoring self shading, for each point x in the image, the illumination at a given point $p_x = (x_1, x_2, u(x_1, x_2))$ of the surface is computed as

$$y(p_x) = \max(0, \langle v(p_x), n(p_x) \rangle) \leqslant 1$$

where $v(p)$ is the unit light vector at point p, $n(p) = \tilde{n}(p)/\|\tilde{n}(p)\|$ is the unit normal at point p, and $\tilde{n}(p_x) = (-\nabla u(x), 1)$.

For an infinite light source, $v(p_x) = v \in \mathbb{R}^3$ is constant and unit normed. A further simplifying assumption is that the camera taking the picture of the object performs an orthographic projection, which is a valid assumption if the object is far away from the optical center. In this case, the intensity $f(x)$ of the image at a pixel x is equal to the illumination $y(p_x)$ at the surface location p_x.

For a vertical light source $v = (0, 0, 1)$, putting together all these simplifying assumptions, one obtains that v satisfies the following Eikonal equation

$$\|\nabla u\| = b(x) = \sqrt{1 - \frac{1}{f(x)^2}}, \tag{3.21}$$

see for instance [175, 242]. This equation is not well defined at singular

points $\Sigma = \{x \setminus f(x) = 1\}$, because $b(x)$ tends to $+\infty$. These singular points Σ correspond to locations where the surface is facing the light direction. The equation can be regularized by replacing $b(x)$ by $\bar{b}(x) = \min(b(x), b_{\max})$.

The solution of the Eikonal shape from shading equation (3.21) should be understood as the unique viscosity solution, subject to some interpolating conditions $f(x_i) = f_i$ for a set of fixed points x_i. These points can be set on the boundary of the object, or at the singular points Σ. This is a major bottleneck of this approach that necessitates some prior knowledge about the object to recover.

Figure 3.31 shows two examples of shape from shading reconstruction using the Fast Marching to solve (3.21). This shows the importance of setting adequate interpolation condition to obtain a valid reconstruction.

The non-uniqueness of shape from shading problem without proper assumptions (such as viscosity solutions and fixed boundary points) reflects the ill-posedness of the problem. These difficulties have been deeply investigated in the literature, and are usually referred to as concave/convex ambiguities, see for instance [18].

For an arbitrary point light source, and a generic perspective camera, Prados and Faugeras have shown in [230] that the shape from shading problem corresponds to solving a more general Hamilton-Jacobi non-linear PDE, which can also be solved with generalized Fast Marching methods.

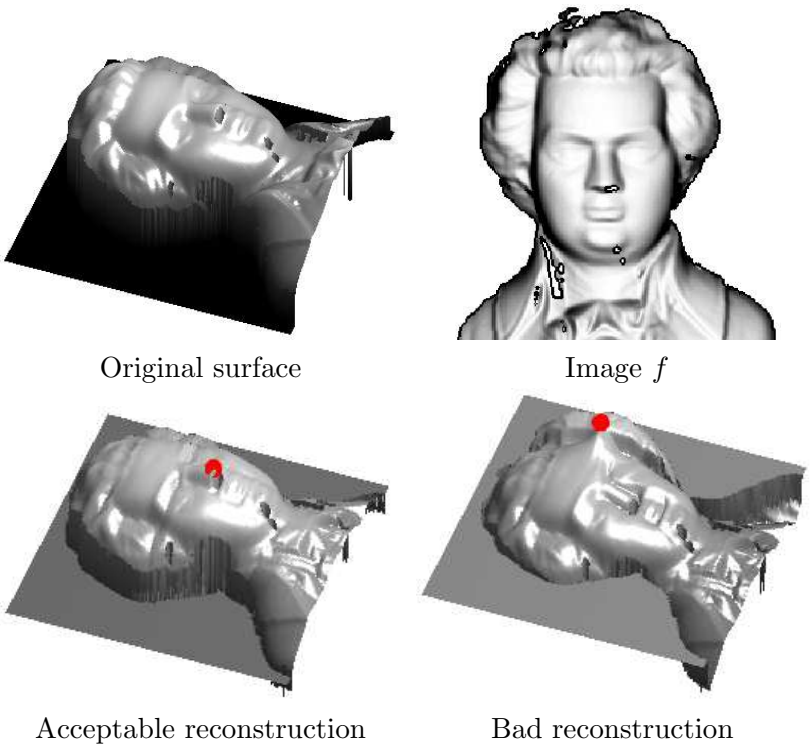


Fig. 3.31 *Bottom row: two examples of shape from shading reconstruction. The red point x_i indicates the location where the condition $f(x_i) = f_i$ is enforced. The original surface is taken from [297].*

4

Geodesic Sampling

In order to acquire discrete samples from a continuous Riemannian manifold, or to reduce the number of samples of finely sampled manifolds, it is important to be able to seed evenly a set of points on a manifold. This is relevant in numerical analysis in order to have a good accuracy in computational simulations, or in computer graphics in order to display 3D models with a low number of polygons. In practice, one typically wants to enforce that the samples are approximately at the same distance from each other according to a given metric. The numerical computation of geodesic distances is thus a central tool, that we are going to use both to produce the sampling and to estimate the connectivity of a triangular mesh.

4.1 Geodesic Voronoi and Delaunay Tesselations

A sampling of a Riemannian manifold Ω is a set of N points $S = \{x_i\}_{i \in I} \subset \Omega$, where $I = \{0, \ldots, N-1\}$. One can compute several topological tesselations on top of this sampling. This section generalizes the notions of Euclidean Voronoi and Delaunay diagrams to arbitrary Riemannian manifolds. In the remaining part of this chapter,

this framework will be used to design efficient sampling schemes.

4.1.1 Voronoi Segmentation

The main geometrical and topological structure associated to a sampling is the Voronoi segmentation, that is at the heart of the computation of the other geodesic tesselations.

Geodesic Voronoi Diagram. When $S = \{x_i\}_{i \in I}$ is finite, one defines a segmentation of the manifold Ω into Voronoi cells as

$$\mathcal{V}(S) = \{\mathcal{C}_i\}_{i \in I} \quad \text{and} \quad \Omega = \bigcup_{i \in I} \mathcal{C}_i \tag{4.1}$$

as defined in (1.18). This segmentation can be represented using the partition function $\ell(x)$ defined in (1.19). Note that the Voronoi cells overlap on their common boundaries.

Geodesic Medial Axis. The medial axis MedAxis(S), defined in section 1.4.2, is the set of points where the distance map U_S is singular. For a dense enough discrete set of points $S = \{x_i\}_{i \in I}$, the medial axis is the boundary of the Voronoi cells, see (1.21).

4.1.2 Delaunay Graph

Delaunay graph. The geodesic Delaunay graph $\mathcal{D}(S)$ of a sampling $S = \{x_i\}_{i \in I} \subset \Omega$ is defined by joining seed points with adjacent Voronoi cells

$$\mathcal{D}(S) = \left\{(i,j) \in I^2 \setminus \partial\mathcal{C}_i \cap \partial\mathcal{C}_j \neq 0\right\}. \tag{4.2}$$

Note that a pair of indices $(i,j) \in \mathcal{D}(S)$ is assumed to be unordered, so that (j,i) denotes the same Delaunay edge.

Geometric realization. For each edge, one can consider its geodesic geometric realization

$$\forall\, (i,j) \in \mathcal{D}(S), \quad \gamma_{i,j} \in \mathcal{P}(x_i, x_j) \tag{4.3}$$

which is the geodesic curve joining x_i and x_j. For a 2D manifold Ω, the Delaunay graph $\mathcal{D}(S)$ is thus a planar graph for this curved realization,

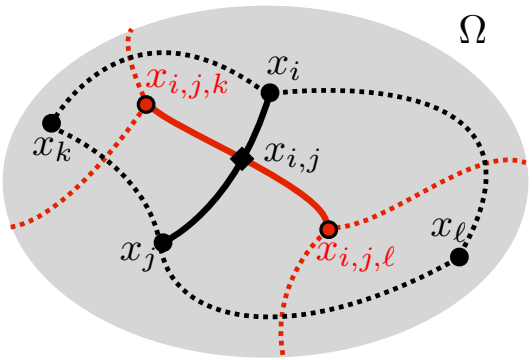


Fig. 4.1 *Schematic diagram of Geodesic and Delaunay.*

which means that the curved edges $\gamma_{i,j}$ do not intersect. This is due to the fact that $\gamma_{i,j} \subset \mathcal{C}_i \cup \mathcal{C}_j$ – see Section 2.5.2 that makes use of this fact to speed up minimal path extraction.

If the manifold is embedded in Euclidean space $\Omega \subset \mathbb{R}^d$ for some $d > 0$, the geodesic realization (4.3) should not be confused with the Euclidean geometric realization $\tilde{\gamma}_{i,j}$, which is the straight line segment

$$\forall\, t \in [0,1], \quad \tilde{\gamma}_{i,j}(t) = (1-t)x_i + t x_j. \tag{4.4}$$

One should note that this straight line embedding of the graph is not necessarily a valid planar embedding, since straight edges $\tilde{\gamma}_{i,j}$ might intersect.

Double saddle points. Each topological Delaunay edge $(i,j) \in \mathcal{D}(S)$ is associated to a dual geometric realization, that is a boundary of Voronoi cells

$$\forall\, (i,j) \in \mathcal{D}(S), \quad \gamma^*_{i,j} = \mathcal{C}_i \cap \mathcal{C}_j.$$

This object is called a dual edge to the primal edge $\gamma_{i,j}$. It is a planar curve for 2D manifolds.

A double point $x_{i,j}$ lies at the intersection of $\gamma_{i,j}$ and $\gamma^*_{i,j}$

$$x_{i,j} = \gamma_{i,j} \cap \gamma^*_{i,j} = \operatorname*{argmin}_{x \in \gamma^*_{i,j}} d(x, x_i), \tag{4.5}$$

as already defined in (2.21). Note that $x_{i,j}$ is not necessarily the point on $\mathrm{MedAxis}(\{x_i, x_j\})$ that is the closest to x_i and x_j, because the dual edge $\gamma^*_{i,j}$ is only a sub-set of $\mathrm{MedAxis}(\{x_i, x_j\})$.

Double point computation. Double points $x_{i,j}$ are computed by processing the boundary of the Voronoi segmentation. This segmentation is computed as detailed in Section 2.6.1.

The geodesic Delaunay edge curve $\gamma_{i,j}$ joining x_i and x_j is extracted by solving two gradient descents to compute the two geodesics joining $x_{i,j}$ to x_i and x_j, as described in Section 2.5.2.

4.1.3 Delaunay triangulation

For simplicity, we restrict ourselves to the setting of 2D manifolds, and consider triangulations of the manifold Ω. Although more difficult to compute, the natural extension to manifolds of dimension d consists of replacing triangles with simplices, which are convex hulls of $d+1$ points.

Triple points. While $\mathcal{D}(S)$ indicates an edge structure based on intersection of pairs of Voronoi cells, it is possible to define a face structure $\mathcal{T}(S)$ by looking at the intersection of three Voronoi cells

$$\mathcal{T}(S) = \{(i,j,k) \setminus \mathcal{C}_i \cap \mathcal{C}_j \cap \mathcal{C}_k \neq \emptyset\}. \tag{4.6}$$

Similarly to Delaunay edge, triple indices are not ordered, and permutations of (i,j,k) denote the same face. For points in generic position, a non-empty intersection of three cells is a triple point

$$\forall\, (i,j,k) \in \mathcal{T}(S), \quad x_{i,j,k} \in \mathcal{C}_i \cap \mathcal{C}_j \cap \mathcal{C}_k. \tag{4.7}$$

For each $(i,j,k) \in \mathcal{T}(S)$, the triple point $x_{i,j,k}$ lies at the intersection of three portions of mediatrix

$$x_{i,j,k} = \gamma^*_{i,j} \cap \gamma^*_{j,k} \cap \gamma^*_{k,i}.$$

It thus corresponds to the geodesic extension of the classical notion of circumcenter in Euclidean geometry.

The boundary of the open geodesic ball of center $x_{i,j,k}$ thus contains three sampling points

$$\{x_i, x_j, x_k\} \subset \partial B_r(x_{i,j,k}) \quad \text{where} \quad B_r(x) = \{y \in \Omega \setminus d(x,y) < r\}$$

for $r = d(x_i, x_{i,j,k})$. The Delaunay triangulation possesses the empty circumcircle property

$$\forall\, \ell \notin \{i,j,k\}, \quad x_\ell \notin B_r(x_{i,j,k}). \tag{4.8}$$

The natural extension of triple points to a manifold of dimension d considers the intersection of d Voronoi cells. This is the geodesic generalization of the ortho-center of a d-dimensional simplex.

Triple point computation. Triple points $x_{i,j,k}$ are computed as a by-product of the extraction of the Voronoi segmentation detailed in Section 2.6.1.

Triangulations. If the metric T_x is a smooth function of x and if the sampling S of Ω is dense enough with respect to the curvature of the manifold, one can prove that the Delaunay graph is equal to the Delaunay triangulation, which means that

$$\forall\, (i,j) \in \mathcal{D}(S),\ \exists k \in I, \quad (i,j,k) \in \mathcal{T}(S),$$

see [165]. The number of points needed for the Delaunay triangulation to be valid depends on the geometry of the manifold, and in particular on its curvature, see [205].

In particular, there are no isolated edges. If the manifold does not have boundary, the Delaunay triangulation defines a valid triangulation of the manifold using the geometric realization (4.3) of the edge.

One can also prove that if the sampling is dense enough, then the straight line realization (4.4) also gives a valid triangulation in the Euclidean space in which the manifold is embedded. This Euclidean triangulation, whose edges are straight segments, is useful for many applications as detailed in Sections 4.2.3, 4.3 and 4.5.

Delaunay/Voronoi geodesic duality. A primal edge $\gamma_{i,j}$ links two (primal) samples $x_i, x_j \in S$, while the corresponding dual edge $\gamma^*_{i,j}$ links (dual) triple points $x_{i,j,k}, x_{i,j,\ell}$. The set of triple points

$$S^* = \{x_{i,j,k} \setminus (i,j,k) \in \mathcal{T}(S)\}$$

thus constitutes a dual sampling of the manifold Ω. These points are connected by dual edges to form polygons which are not necessarily triangles.

Euclidean Voronoi and Delaunay. The special case of the Euclidean metric $T_x = \mathrm{Id}_2$ in $\Omega = \mathbb{R}^2$ has been extensively studied. The Delaunay triangulation [91] of a sampling S is a valid triangulation of the convex hull of S. It is furthermore unique for points in generic positions. It is characterized by the fact that the circumcircle of a triangle (x_i, x_j, x_k) for $(i, j, k) \in \mathcal{T}$ does not contain any other point, which corresponds to condition (4.8). There exist several iterative algorithms to find the Delaunay triangulation of a set of N points in $O(N \log(N))$ operations, see for instance [88].

Figure 4.2 shows an example of Euclidean Delaunay triangulation.

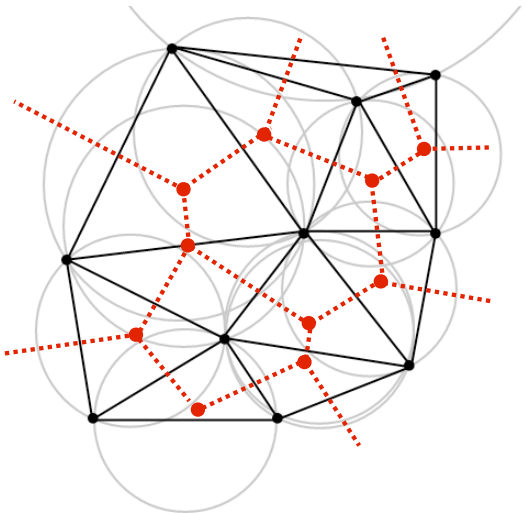


Fig. 4.2 *Example of Voronoi diagram (dashed) and Delaunay triangulation for the Euclidean metric, with the circumcenter in light gray.*

4.2 Geodesic Sampling

The Riemannian metric T_x is used to control the quality of a sampling $\{x_i\}_{i\in I} \subset \Omega$. Finding a sampling with high quality corresponds to finding a sampling whose density and anisotropy conform to the metric, and is useful in many applications, ranging from finite element simulations to computer graphics.

4.2.1 Riemannian Sampling Constraint

Covering and packing sampling constraints. For a given sampling distance $\varepsilon > 0$, one looks for a sampling such that all pairs of neighboring points are approximately the same distance apart ε. The notion of neighbors is, however, difficult to define. Following [53], one can replace it by looking at geodesic balls of radius ε, already introduced in equation (3.20)

$$B_\varepsilon(x) \stackrel{\text{def.}}{=} \{y \setminus d(x,y) \leqslant \varepsilon\}.$$

A sampling $S = \{x_i\}_{i\in I} \subset \Omega$ is an ε-covering, for some $\varepsilon > 0$ if

$$\bigcup_{i\in I} B_\varepsilon(x_i) = \Omega, \tag{4.9}$$

which means that any point $x \in \Omega$ is at a geodesic distance less than ε from S, or equivalently that $U_S \leqslant \varepsilon$. Figure 4.3, left, shows an example of ε-covering.

To forbid such an ε-sampling to contain too many points, one requires that it is an η-packing in the sense that

$$\forall\, i,j \in I, \quad i \neq j \quad \Longrightarrow \quad d(x_i,x_j) \geqslant \eta \tag{4.10}$$

which means that balls of radius $\eta/2$ centered at points in S do not overlap. Figure 4.3, middle, shows an example of ε-packing.

An ε-net is a sampling that is both an ε-covering and an ε-packing. Figure 4.3, right, shows an example of ε-net. Those sets are also called Delone sets in [67], and they can be shown to have optimality properties for the approximation of functions defined on Ω. Searching for an ε-covering that is an η-packing for the largest η corresponds to

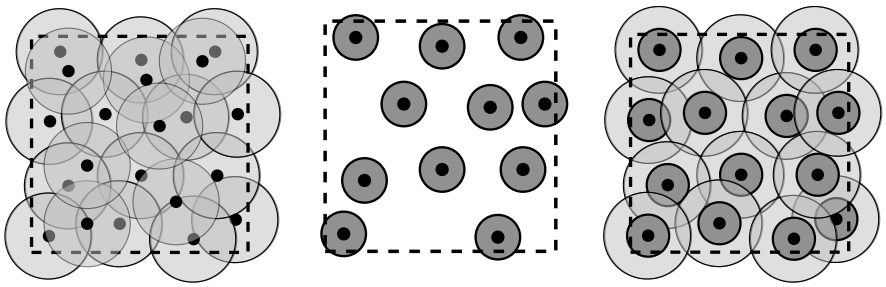


Fig. 4.3 *Comparison of the packing and covering properties for a Euclidean square (dashed lines). The light gray circle are Euclidean balls of radius ε while dark circle are balls of radius $\varepsilon/2$.*

the problem of (geodesic) sphere packing, and is a deep mathematical problem even in Euclidean space [79].

An efficient ε-net should contain the smallest possible number N of points. Finding a sampling that satisfies these conditions with the smallest N is a difficult problem. A simple greedy procedure to approximate this problem, proposed originally in [125], constructs iteratively an ε-net $\{x_i\}_{i\in I}$. It starts by some random point $x_0 \in \Omega$ and then iteratively adds a new point at random that satisfies

$$x_{k+1} \in \Omega \backslash \bigcup_{i=0}^{k} B_\varepsilon(x_i), \tag{4.11}$$

until condition (4.9) is enforced.

Using a random choice in the greedy process (4.11) usually leads to a poor sampling quality so that N can be quite large. Section 4.2.2 details a non-random selection process that usually leads to a good solution.

Distance conforming Riemannian sampling. A way to make more explicit the control of the sampling by the metric is to use the Delaunay graph (4.2) as a notion of neighborhood. The sampling is said to be distance conforming to the metric if

$$\forall\, (i,j) \in \mathcal{D}(S), \quad C_1\varepsilon \leqslant d(x_i,x_j) \leqslant C_2\varepsilon \tag{4.12}$$

where ε is the sampling precision, and C_1, C_2 are constants independent of N.

Note that if S is an ε-net, and thus satisfies (4.10) and (4.9), then (4.12) is satisfied for $d(x_i, x_j) \in [\varepsilon, 2\varepsilon]$.

Isotropic metric and density sampling. In the case of an isotropic metric $T_x = W(x)^2 \mathrm{Id}_d$ in $\Omega \subset \mathbb{R}^d$, the sampling constraint becomes

$$\forall\, (i,j) \in \mathcal{D}(\Omega), \quad \|x_i - x_j\| \approx \frac{\varepsilon}{W(x_i)}, \tag{4.13}$$

Under these conditions, in a Euclidean ball of radius $r > 0$ centered at x, the number of samples should be proportional to $r^d W(x)^d$. The constraint thus corresponds to imposing that the sampling density is proportional to $W(x)^d$. In particular, regions of Ω where W is high are constrained to use a dense sampling.

One should note that this density sampling requirement does not correspond to drawing point at random according to the density $W(x)/\int_\Omega W$, since one wishes to have neighbording points which conform as much as possible to the metric, which random sampling usually does not achieve.

In 1D, if $\Omega = [0,1]$, the isotropic sampling problem is easily solved. A perfect sampling conforming to the metric $W(x)$ is defined as

$$x_i = F^{-1}(i/N) \quad \text{where} \quad F(x) = \frac{1}{\int_0^1 W} \int_0^x W(y)\mathrm{d}y. \tag{4.14}$$

Obtaining a good density sampling for 2D and higher dimensional manifolds is difficult. A simple greedy procedure is the error diffusion method [118] and extensions [210], which is mainly used for digital halftoning [15]. This method operates on a uniform grid, and scans in a given order the grid cells to reduce the sampling problem to a 1D repartition problem, similarly to (4.14).

Other approaches, based on irregular planar tilings, offer better performances without periodic artifacts [211, 212].

The following section details a greedy sampling procedure that can produce a good sampling in practice, and can take into account anisotropic Riemannian metrics.

Triangulation conforming Riemannian sampling. The sampling condition (4.13) only constrains the length of the Delaunay edges. For many applications, including the resolution of elliptic PDEs, and the approximation of smooth images (see Section 4.3), it is also required that Delaunay triangles in $\mathcal{T}(S)$ are close to being equilateral, when seen from the metric T_x. Roughly speaking, if $\triangle \in \mathcal{T}$ is a triangle centered around $x_0 \in \triangle$, it should be enclosed in two concentric ellipsoids

$$\big\{x \setminus \|x - x_0\|_{T_{x_0}} \leqslant C_0\varepsilon\big\} \subset \triangle \subset \big\{x \setminus \|x - x_0\|_{T_{x_0}} \leqslant C_1\varepsilon\big\} \qquad (4.15)$$

where ε controls the number N of samples and C_0, C_1 are two constants. Note that the distance constraint (4.13) does not ensure that the triangles have approximately equal edges, as shown on figure 4.4, center.

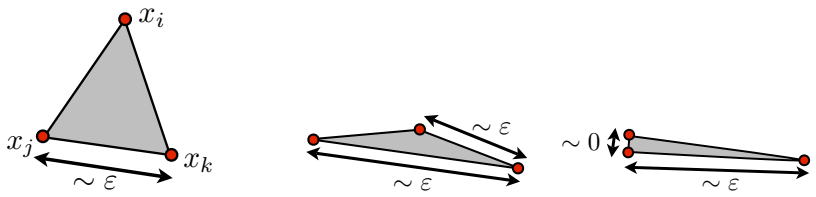


Fig. 4.4 *Shapes of triangles with distance and/or triangulation conforming to the Euclidean constant metric.*

4.2.2 Farthest Point Sampling

The farthest point sampling algorithm is a greedy strategy able to produce quickly a good sampling which turns out to be an ε-net. Instead of performing a random choice in (4.11), it selects the farthest point from the already selected points

$$x_{k+1} = \operatorname*{argmax}_{x \in \Omega} \min_{0 \leqslant i \leqslant k} d(x_i, x). \qquad (4.16)$$

This selection rule first appeared in [125] as a clustering method, see also [87] for an analysis of clustering algorithms. This algorithm has

been introduced in image processing to perform image approximation in [112]. It was used as well in [77] to perform perceptual grouping through the iterative adding of key-points and detection of saddle points (equivalent to the double saddle points above for the geodesic distance). It was then extended in [221] together with geodesic Delaunay triangulation to do surface remeshing.

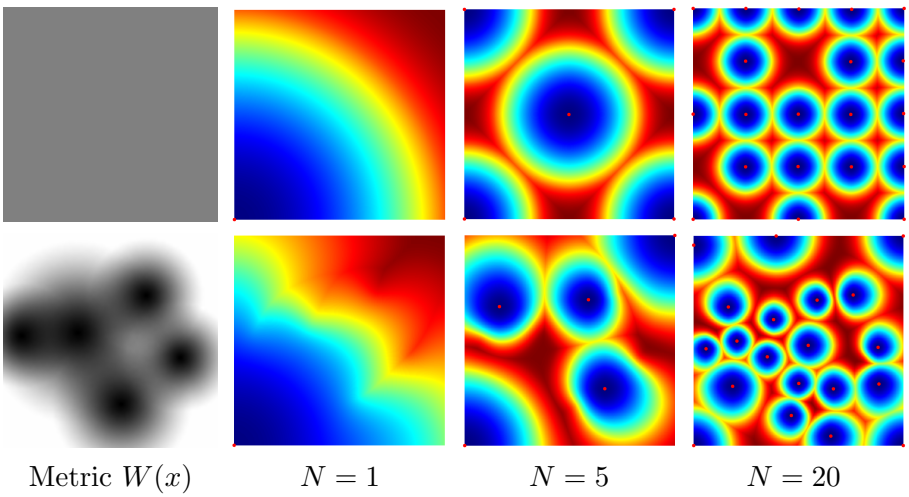


Fig. 4.5 *Examples of farthest point sampling, the colormap indicates the distance function U_S.*

Figure 4.5 shows some iterations of this farthest point sampling method for isotropic metrics on a square. One can see that this scheme seeds more points in areas where the metric W is large. One can thus control the sampling density by modifying the metric W. Algorithm 7 gives the detail of the algorithm.

Algorithm 7: Farthest point sampling algorithm.

Initialization: set x_0 at random, $d_0(x) = d(x_0, x)$, $k = 0$.

while $\varepsilon_k > \varepsilon$ **do**

  Select point: $x_{k+1} = \operatorname{argmax}_x d_k(x)$, $\varepsilon_{k+1} = d_k(x_{k+1})$.

  Distance update: $\forall x \; d_{k+1}(x) = \min(d_k(x), d(x_{k+1}, x))$.

  Set $k \leftarrow k + 1$.

Numerical complexity. Denoting

$$d_k(x) = \min_{0 \leqslant i \leqslant k} d(x_i, x) = U_{\{x_0,\ldots,x_k\}}(x),$$

the selection rule (4.16) reads

$$x_{k+1} = \operatorname*{argmax}_{x \in \Omega} d_k(x),$$

while d_{k+1} is computed from d_k as

$$d_{k+1}(x) = \min(d_k(x), d(x_{k+1}, x)).$$

This update of the distance map is performed efficiently by a single Fast Marching propagation, starting from x_{k+1}, and restricted to the Voronoi region of x_{k+1}

$$\mathcal{C}_{k+1} = \{x \in \Omega \setminus \forall\, i \leqslant k,\ d(x_{k+1}, x) \leqslant d(x_i, x)\}\,.$$

If the manifold is discretized with N_0 points and if the metric T_x does not vary too much, the size of $\mathcal{C}_{k+1}$ is roughly $O(N_0/k)$. Hence the complexity of each sampling step is $O(N_0/k \log(N_0))$, and the overall complexity of sampling $N \ll N_0$ points is roughly $O(N_0 \log(N_0) \log(N))$.

Farthest sampling quality. The farthest point sampling $\{x_0, \ldots, x_{N-1}\}$ is an ε-net for

$$\varepsilon = \max_{0 \leqslant i < N} \min_{0 \leqslant j < N} d(x_i, x_j). \tag{4.17}$$

Note however that there is no simple control on the number of samples N required to achieve a given accuracy ε. We refer to [67] for an in-depth study of the approximation power of this greedy sampling scheme.

4.2.3 Farthest Point Meshing

This Section considers the particular case of 2D manifolds $\Omega \subset \mathbb{R}^2$, or 2D surfaces embedded in Euclidean space. We also restrict ourself to the case of manifolds without boundaries. Special care is required to correctly approximate the boundary of the manifold, see Section 4.5.

Having computed a sampling $\{x_i\}_{i\in I} \subset \Omega$, one can define a triangulation of the manifold Ω using the geodesic Delaunay faces $\mathcal{T}(S)$ defined in (4.6). One can connect the samples using the geodesic curve realization (4.3) or using the straight line realization (4.4) if the manifold is embedded in Euclidean space.

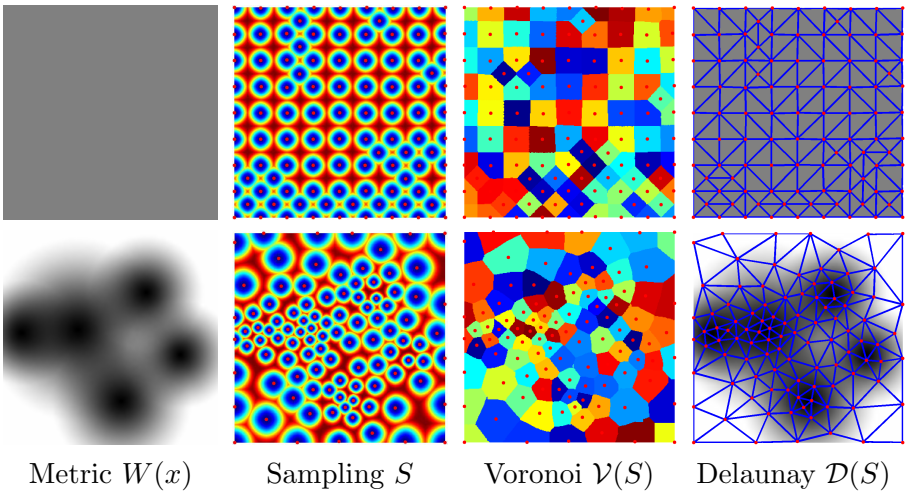


Fig. 4.6 *Examples of sampling and triangulations with an isotropic metric* $T_x = W(x)^2 \mathrm{Id}_2$. *The sampling is denser in the regions where the metric is larger (dark).*

The resulting mesh can be used to mesh a continuous domain or re-mesh a densely sampled manifold as explained in [220]. Figure 4.6 shows the process of computing the sampling, the Voronoi regions, and the Delaunay triangulation.

Sections 4.3, 4.4 and 4.5 show applications of this algorithm to the meshing of images, surfaces and sub-domains.

Triangulation validity and metric gradation. The farthest point sampling $\{x_0, \ldots, x_{N-1}\}$ is distance conforming, and (4.13) holds for $C_1 = 1, C_2 = 2$ and ε defined in (4.17). For the sampling to be triangulation conforming and satisfy (4.15), the metric T_x should not exhibit strong variations. For an isotropic metric $T_x = W(x)^2 \mathrm{Id}_2$, the sizing field $W(x)^{-1}$ should be 1-Lipshitz to ensure triangulation con-

formance, and this gradation condition extends to anisotropic metric, see [158]. If the metric T_x exhibits strong variations, it is mostly an open question how to smooth it so that the modified metric is graded, although heuristics have been proposed, see for instance [222, 7] for isotropic gradation and [170, 35, 4] for anisotropic gradation. To mesh the interior of a domain, a valid graded metric can be defined from a boundary metric using (4.48).

4.3 Image Meshing

In this section, we consider the case of a 2D manifold parameterized on a square so that $\Omega = [0,1]^2$. The goal is to use the Riemannian structure to perform image sampling and triangulation.

It is possible to use a user defined metric T_x to drive the sampling, as shown in Section 4.3.1. One can also design the tensor field to minimize the approximation error of an image $f(x)$ using a spline approximation on the geodesic triangulation. In this case, the eigenvectors $e_1(x)$ defined in (1.16) should match the direction of edges and textures in the image, while the anisotropy $A(x)$, defined in (1.17), should match the anisotropic regularity of f near x.

4.3.1 Density Meshing of Images

A geodesic isotropic triangulation with $T_x = W(x)^2 \mathrm{Id}_2$ seeds points according to a density function $W(x)^2$ for $x \in [0,1]^2$. Regions where $W(x)$ is larger get more samples.

Figure 4.7 shows triangulations obtained for several isotropic metrics. It shows how the triangulation is refined as the farthest point algorithm inserts new samples.

4.3.2 Image Approximation with Triangulations

Adaptive image approximation is performed by computing a triangulation $\mathcal{T}$ of the image and using a piecewise linear finite elements approximation. This class of methods originates from the discretization of partial differential equations, where the design of the elements should match the regularity one expects for the solutions, which might contain

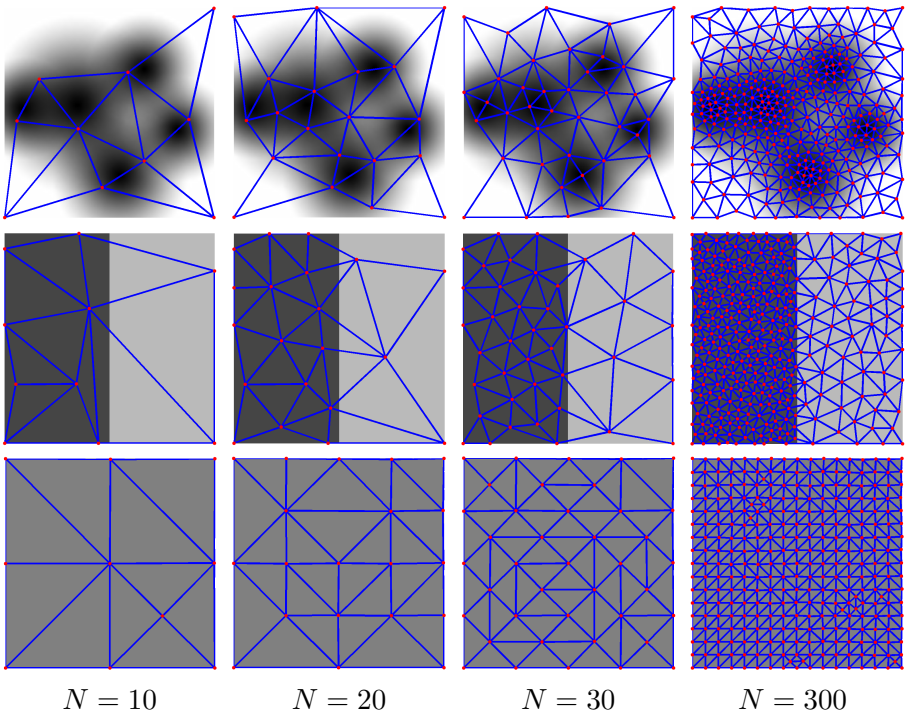


Fig. 4.7 *Examples of farthest point meshing for different isotropic metrics $W(x)$ (shown in background) and different values of the number of samples N.*

shocks or boundary layers. Iterative adaptation allows to refine both the solution of the equation and the shape of the elements [2, 237, 259].

The positions of the samples $S = \{x_0, \dots, x_{N-1}\}$ and the connectivity of the triangulation $\mathcal{T}$ should be adapted to the features of the image. Note that, in general, $\mathcal{T}$ is not necessarily a Euclidean or geodesic Delaunay triangulation $\mathcal{T}(S)$ of S. In the following, to ease the explanations, we consider $\mathcal{T}$ as a collection of triangles $t \in \mathcal{T}$, and not sets of indexes.

A piecewise affine function f_N on the triangulation is defined as

$$f_N = \sum_{i \in I} a_i \varphi_i,$$

where φ_i is the hat spline function, that is affine on each triangle and such that $\varphi_i(x_j) = 0$ for $i \neq j$ and $\varphi_i(x_i) = 1$.

The efficiency of the approximation f_N is measured using the L^p

norm on the domain, for $1 \leqslant p \leqslant +\infty$

$$\|f - f_N\|_{L^p(\Omega)}^p = \int_\Omega |f(x) - f_N(x)|^p \mathrm{d}x \tag{4.18}$$

and

$$\|f - f_N\|_{L^\infty(\Omega)} = \max_{x \in \Omega} |f(x) - f_N(x)|. \tag{4.19}$$

It is possible to use an interpolation of the original image by defining $a_i = f(x_i)$. If one measures the approximation error using the L^2 norm, a better approximation is obtained by an orthogonal projection

$$f_N = \sum_{i \in I} a_i \varphi_i \quad \text{where} \quad a = \operatorname*{argmin}_{\tilde{a} \in \mathbb{R}^N} \|f - \sum_i \tilde{a}_i \varphi_i\|^2. \tag{4.20}$$

The coefficient a of this approximation f_N is computed by solving a sparse linear system

$$\forall\, i \in I, \quad \sum_j \langle \varphi_i, \varphi_j \rangle a_j = \langle f, \varphi_i \rangle.$$

4.3.3 Greedy Schemes

Given a fixed number N of vertices, the goal is to design a triangulation so that the approximation error $\|f - f_N\|_{L^p(\Omega)}$ is as low as possible. Such an efficient triangulation is likely to be also efficient for applications to image compression and denoising, because it captures well the geometry of the image.

Computing this optimal triangulation is in some sense NP-hard [1], and one thus needs to rely on sub-obtimal greedy schemes. These schemes generate a sequence of triangulations by either refinement (increasing N) or coarsening (decreasing N, starting from a dense sampling).

Refinement schemes. A greedy refinement scheme starts by a simple fixed triangulation $(\mathcal{T}_0, S_0)$ of the squares $[0,1]^2$, and iteratively adds one or several vertices to S_j to obtain a triangulation $(\mathcal{T}_{j+1}, S_{j+1})$ that minimizes the approximation error.

The Delaunay refinement introduced by Ruppert [245] and Chew [63], proceed by inserting a single point, which is a circumcenter of one triangle. One also imposes that $\mathcal{T}_j = \mathcal{T}(S_j)$ is a Delaunay

triangulation of S_j. This constraint limits the domain of the optimization and thus accelerates the search, and also leads to triangles with provably good isotropic aspect ratio, which might be useful to compute an approximation of the solution of an elliptic PDE on the mesh grid. For image approximation, one however needs to design anisotropic triangulations. This requires to modify the notion of circumcenter, using an anisotropic metric [123]. Other refinements are possible, such as for instance edge bisection [190], that reaches the optimal asymptotic error decay for smooth convex functions.

Coarsening schemes. Triangulation coarsening algorithms start with a fine scale triangulation $(\mathcal{T}_J, S_J)$ of $[0,1]^2$ and progressively remove either a vertex, an edge or a face to increase the approximation error as slowly as possible until N vertices remain [110, 134, 122]. One can for instance remove a single vertex to go from S_{j+1} to S_j, and impose that $\mathcal{T}_j = \mathcal{T}(S_j)$ is the Delaunay triangulation of S_j. This can be shown experimentally to produce highly anisotropic meshes, which can be used to perform compression, see [92].

4.3.4 Hessian Tensor Metric

In this section, we consider a uniformly smooth C^2 image defined on Ω. We show how to design locally a metric T_x so that if the triangulation conforms to this metric, $\|f - f_N\|_{L^p(\Omega)}$ is as small as possible.

Local error optimization. Near a point $x \in \Omega$, the error obtained when approximating f with an affine function is governed by the Hessian matrix H_f of second derivatives :

$$H_f(x) = \left(\frac{\partial^2 f}{\partial x_i \partial x_j}(x) \right)_{0 \leqslant i,j \leqslant 1}. \tag{4.21}$$

One can diagonalize this symmetric matrix field as follows

$$H_f(x) = \lambda_1(x) e_1(x) e_1(x)^{\mathrm{T}} + \lambda_2(x) e_2(x) e_2(x)^{\mathrm{T}}, \tag{4.22}$$

where (e_1, e_2) are the orthogonal eigenvectors fields and $|\lambda_1| \geqslant |\lambda_2|$ are the eigenvalues fields.

In the following, we assume that H_f does not vary too much so that it can be considered as constant inside each triangle $t \in \mathcal{T}$. This heuristic can be made rigorous, see for instance [190].

Performing a Taylor expansion of f near a vertex $x_k \in t$ for a triangle $t \in \mathcal{T}$ of the triangulation leads to

$$|f(x) - f_N(x)| \leqslant C|\lambda_1(x_k)||\langle x - x_k,\, e_1(x_k)\rangle|^2 \tag{4.23}$$

$$+ C|\lambda_2(x_k)||\langle x - x_k,\, e_2(x_k)\rangle|^2. \tag{4.24}$$

where C is a constant that does not depend on N. Denoting as $\Delta_1(x_k)$ and $\Delta_2(x_k)$ the size of the triangle t in each direction $e_1(x_k)$ and $e_2(x_k)$, one obtains the pointwise error bound

$$|f(x) - f_N(x)| = O\left(|\lambda_1(x_k)|\Delta_1(x_k)^2 + |\lambda_2(x_k)|\Delta_2(x_k)^2\right). \tag{4.25}$$

Uniform triangulation. For a uniform triangulation, where all the triangles $t \in \mathcal{T}$ are approximately equilateral with the same size, one has

$$\Delta_1(x) \approx \Delta_2(x) \approx N^{-1/2},$$

so that the approximation error in (4.25) leads to

$$\|f - f_N\|_{L^p(\Omega)} \leqslant C\|H_f\|_{L^p(\Omega)} N^{-1} \tag{4.26}$$

where the L^p norm of the Hessian field is

$$\|H_f\|^p_{L^p(\Omega)} = \int_\Omega |\lambda_1(x)|^p \mathrm{d}x.$$

Isotropic triangulation. An isotropic triangulation makes use of triangles that are approximately equilateral, so that $\Delta_1(x) \approx \Delta_2(x)$, and the error (4.25) leads on each triangle $t \in \mathcal{T}$ to

$$\|f - f_N\|_{L^p(t)} \leqslant C\|H_f\|_{L^q(t)} \quad \text{where} \quad \frac{1}{q} = 1 + \frac{1}{p}.$$

In order to reduce as much as possible the approximation error $\|f - f_N\|_{L^p(\Omega)}$ on the whole domain, a heuristic is to equidistribute the approximation error on all the triangles. This heuristic can be shown to be nearly optimal, see [190]. This criterion requires that for $x_k \in t \in \mathcal{T}$,

$$\|H_f\|_{L^q(t)} \approx |t|^{1/q}|\lambda_1(x_k)| \approx \Delta_1(x_k)^{2/q}|\lambda_1(x_k)| \tag{4.27}$$

is approximately constant, where $|t|$ is the area of the triangle. This means that the triangle t located near x_k should have approximately constant edge size, for the isotropic Riemannian metric T_{x_k} defined as

$$T_x = W(x)^2 \mathrm{Id}_2 \quad \text{where} \quad W(x)^2 = |\lambda_1(x)|^q. \tag{4.28}$$

For instance, if one measures the approximation error using the L^∞ norm, then $W(x)^2 = |\lambda_1(x)|$. An adaptive isotropic triangulation conforming to the metric (4.28), so that (4.15) holds, gives rise to an approximation error

$$\|f - f_N\|_{L^p(\Omega)} \leqslant C \|H_f\|_{L^q(\Omega)} N^{-1} \quad \text{where} \quad \frac{1}{q} = 1 + \frac{1}{p}. \tag{4.29}$$

Since $q < p$, note that the constant appearing in the isotropic approximation (4.29) is much smaller than the constant in the constant size approximation (4.26).

Anisotropic triangulation. As detailed in [14], for a smooth function, one should use anisotropic triangles whose aspect ratio match the anisotropy of the image. To reduce as much as possible the point-wise error (4.23), the error along each axis e_1, e_2 should be approximately equal, so that the anisotropy of the triangles should satisfy

$$\frac{\Delta_1(x)}{\Delta_2(x)} = \sqrt{\frac{|\lambda_2(x)|}{|\lambda_1(x)|}}. \tag{4.30}$$

Under this anisotropy condition, the error (4.23) leads on each triangle $t \in \mathcal{T}$ to

$$\|f - f_N\|_{L^p(t)} \leqslant C \left\| \sqrt{|\det(H_f)|} \right\|_{L^q(t)} \quad \text{where} \quad \frac{1}{q} = 1 + \frac{1}{p},$$

see [190]. Similarly to the isotropic case (4.27), the equidistribution of error criterion leads to

$$\left\| \sqrt{|\det(H_f)|} \right\|_{L^q(t)} \approx |t|^{1/q} \sqrt{|\lambda_1(x_k)\lambda_2(x_k)|} \tag{4.31}$$

$$\approx (\Delta_1(x_k)\Delta_2(x_k))^{1/q} \sqrt{|\lambda_1(x_k)\lambda_2(x_k)|} \tag{4.32}$$

being approximately constant.

Conditions (4.30) and (4.31) show that a triangle of an optimal triangulation for the L^p norm should have its edges of equal length when measured using the following Riemannian metric

$$T_x = |\det(H_f(x))|^{\frac{q-1}{2}} |H_f(x)| \tag{4.33}$$

where the absolute value of the Hessian is

$$|H_f(x)| = |\lambda_1(x)| e_1(x) e_1(x)^{\mathrm{T}} + |\lambda_2(x)| e_2(x) e_2(x)^{\mathrm{T}}.$$

For instance, when using the L^∞ norm, the metric is $T_x = |H_f(x)|$. Note that when $p < \infty$, the metric (4.33) is singular at points x where $\det(H_f(x)) = 0$. This can be avoided numerically by using

$$T_x = (|\det(H_f(x))| + \varepsilon)^{\frac{q-1}{2}} |H_f(x)|$$

for a small $\varepsilon > 0$.

An adaptive anisotropic triangulation conforming to the metric (4.33), so that (4.15) holds, gives rise to an approximation error

$$\|f - f_N\|_{L^p(\Omega)} \leqslant C \|\sqrt{|\det(H_f)|}\|_{L^q(\Omega)} N^{-1} \quad \text{where} \quad \frac{1}{q} = 1 + \frac{1}{p}. \tag{4.34}$$

Note that the constant appearing in the anisotropic approximation (4.34) is much smaller than the constant in the isotropic approximation (4.29).

Farthest point Hessian triangulation. Equations (4.28) and (4.33) give respectively the optimal isotropic and anisotropic Riemannian metric that should be used to design triangulations in order to approximate smooth functions. One can thus use the farthest point meshing algorithm detailed in Section 4.2.3 to compute an ε-net that conforms to this metric.

Figure 4.8 shows the evolution of the meshing algorithm for the anisotropic metric (4.33) for the L^∞ norm. Figure 4.9 shows a comparison of the isotropic and anisotropic metrics. One can see the improvement brought by adaptivity and anisotropy.

4.3.5 Structure Tensor Metric

The optimal Hessian-based metrics (4.28) and (4.33) are restricted to the approximation of smooth images. Furthermore, these metrics are

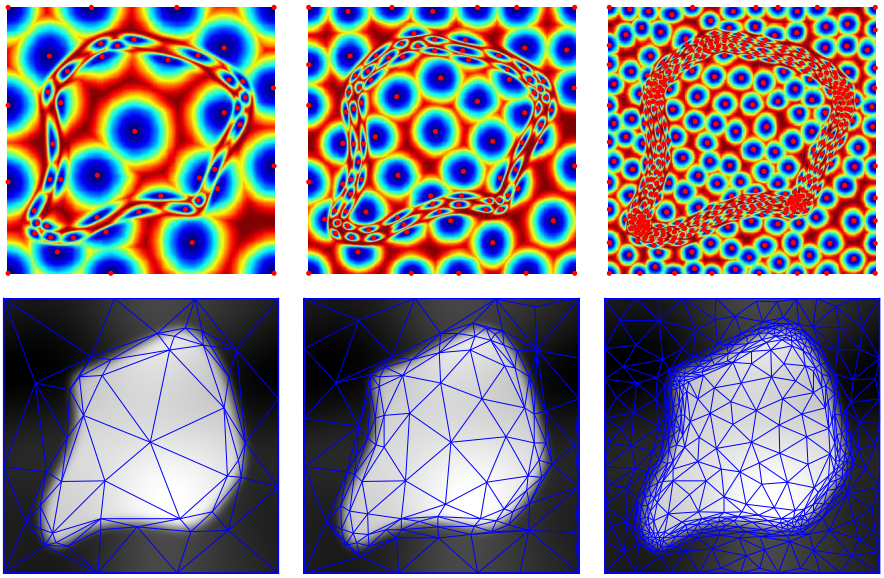


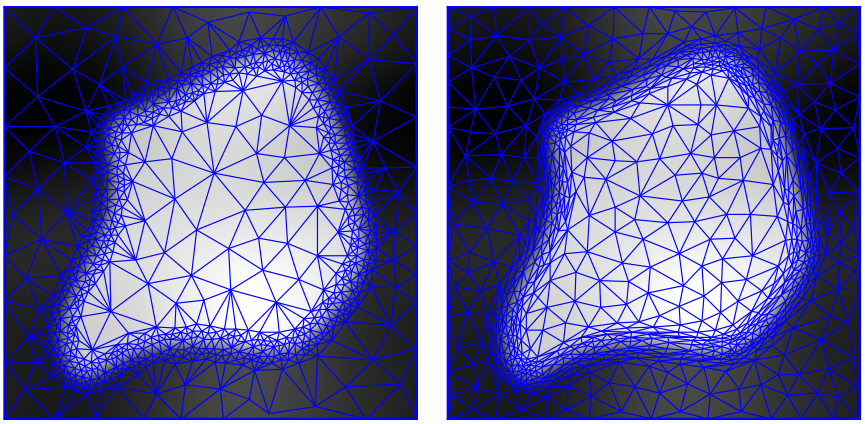
Fig. 4.8 *Geodesic image meshing with an increasing number of points N, for the curvature-driven metric defined in* (3.16)*. Top row: geodesic distance U_S, bottom row: geodesic Delaunay triangulation $\mathcal{D}(S)$.*

quite unstable since second order derivatives are difficult to estimate on noisy images.

To approximate images with step edges, or noisy images, the computation of the optimal metric requires a prior smoothing of the image, and the amount of smoothing depends on the noise level and the number of samples N. Coarse approximation corresponding to a small value of N or a large noise level requires a larger smoothing kernel.

An alternative method computes a robust estimation of both edges and textures directions from first order derivatives using the so-called structure tensor. There is no optimality result for approximation using such first order metrics, but they show good results for image approximation [37].

Structure tensor. The local orientation of a feature around a pixel x is given by the vector orthogonal to the gradient $v(x) = \nabla f(x)$, which



Isotropic $T_x = \|H_f(x)\|\mathrm{Id}_2$ Anisotropic $T_x = |H_f(x)|$

Fig. 4.9 *Comparison of isotropic and anisotropic triangulation of a C^2 image, with $N = 800$ points.*

is computed numerically with finite differences. This local direction information can be stored in a rank-1 tensor $\tilde{T}(x) = v(x)v(x)^{\mathrm{T}}$. In order to evaluate the local anisotropy of the image, one needs to average this tensor

$$T(x) = \tilde{T} \star G_\sigma(x) \tag{4.35}$$

where the 4 entries of the tensor are smoothed against a gaussian kernel G_σ of width $\sigma > 0$. The metric T corresponds to the so-called structure tensor, see for instance [156]. This local tensor T is able to extract both the local direction of edges and the local direction of textural patterns.

At each pixel location x, the structure tensor field can be diagonalized in an orthogonal basis (e_1, e_2)

$$T(x) = \mu_1(x)e_1(x)e_1(x)^{\mathrm{T}} + \mu_2(x)e_2(x)e_2(x)^{\mathrm{T}}, \tag{4.36}$$

where $\mu_1 \geqslant \mu_2 \geqslant 0$. In order to turn the structure tensor $T(x)$ into a Riemannian metric T_x, one can modify the eigenvalues using increasing mappings ψ_i,

$$T_x = \psi_1(\mu_1(x))e_1(x)e_1(x)^{\mathrm{T}} + \psi_2(\mu_2(x))e_2(x)e_2(x)^{\mathrm{T}}. \tag{4.37}$$

for instance $\psi_i(a) = (\varepsilon + a)^\beta$ for a small value of ε and some $\beta > 0$. The parameter ε controls the isotropic adaptivity of the metric, while

β controls the overall anisotropy. A well chosen set of parameters (ε, β) allows one to enhance the resulting image approximation scheme.

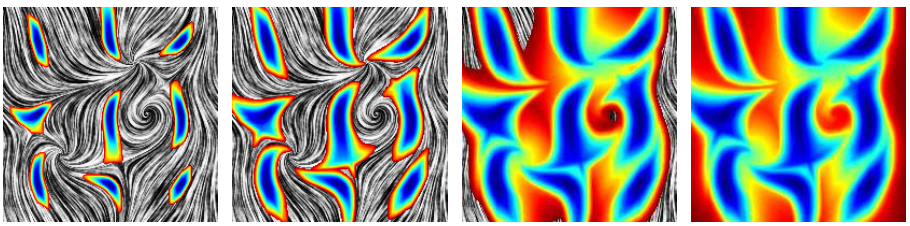


Fig. 4.10 *Examples of anisotropic front propagation (from 9 starting points). The colormap indicates the values of the distance functions at a given iteration of the algorithm. The metric is computed using the structure tensor, equation* (4.35)*, of the texture f shown in the background.*

Figure 4.10 shows an example of Fast Marching propagation using an anisotropic metric T_x computed using the structure tensor.

Anisotropic geodesic meshing for image compression. It is possible to use anisotropic triangulations to perform image compression. This requires to quantize and code the positions of the vertices $\{x_i\}_{i\in I}$ and the value of $\{f_N(x_i)\}_{i\in I}$. Optimizing the distortion rate of the resulting code is difficult because of the lack of orthogonality of the spline approximation, so one has to use heuristics to derive quantization rules.

Figure 4.11 shows an example of image coding with a geodesic triangulation, see [37] for more details about the coding process.

4.4 Surface Meshing

The farthest point sampling algorithm can be used on a surface $\mathcal{S} \subset \mathbb{R}^3$ represented by a discrete 3D mesh that is densely sampled. The method thus performs a sub-sampling followed by a geodesic remeshing of the original triangulated surface.

The density and anisotropy of the final mesh is controlled by a metric $T_{\tilde{x}}$ defined on the tangent plane $\mathcal{T}_{\tilde{x}}$ of the surface $\mathcal{S}$, as introduced in Section 2.4.1. The resulting adaptive mesh can be tuned by the user

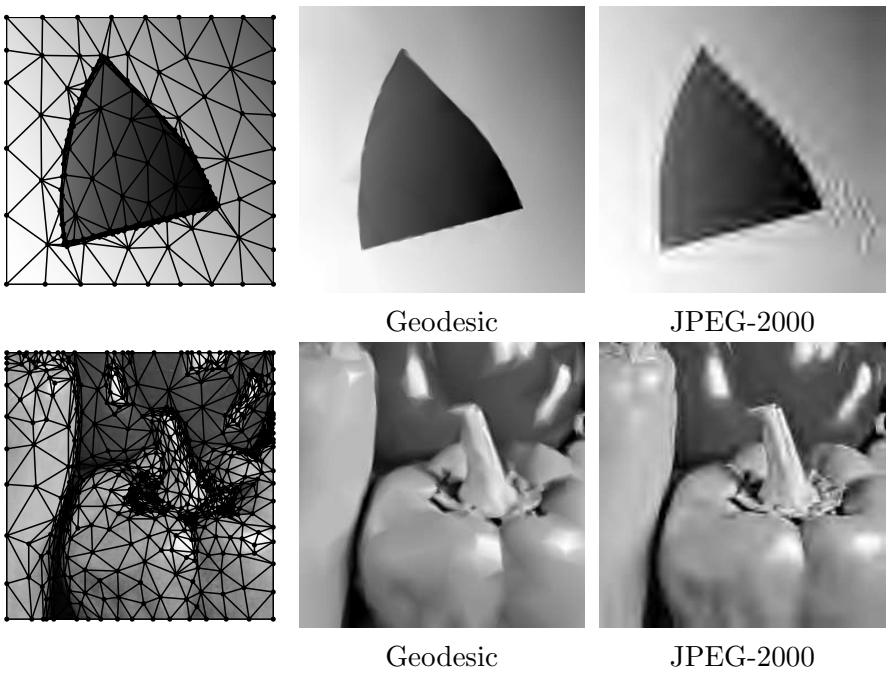


Fig. 4.11 *Comparison of the adapted triangulation scheme [37] with JPEG-2000, for the same number of bits, for $N = 200$ (top) and $N = 600$ (bottom) vertices.*

using a metric computed from a texture map or from the curvature tensor.

4.4.1 Density Meshing of Surfaces

A geodesic isotropic triangulation with $T_{\tilde{x}} = W(\tilde{x})^2 \mathrm{Id}_2$ seeds points according to a density function $W(\tilde{x})^2$ for $\tilde{x} \in \mathcal{S}$.

Figure 4.12 shows an example of uniform remeshing of a surface $\mathcal{S} \in \mathbb{R}^3$ acquired from medical imaging with an increasing number of points, with a constant metric $W(\tilde{x}) = 1$ for $\tilde{x} \in \mathcal{S}$.

Figure 4.13 shows an example of uniform remeshing of the David surface, where the original input surface was obtained by range scanning [166].

As explained in Section 3.2.4, one can define a varying density $W(\tilde{x})$ on the surface. This allows to obtain an adaptive isotropic remeshing of the surface. Figure 4.14 shows how a varying metric (bottom row)

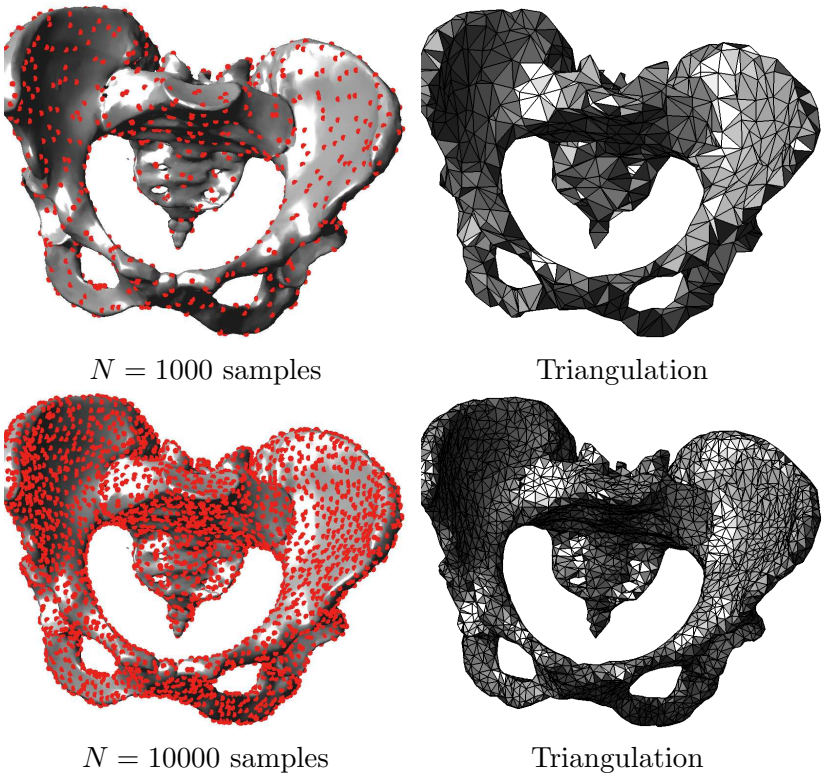
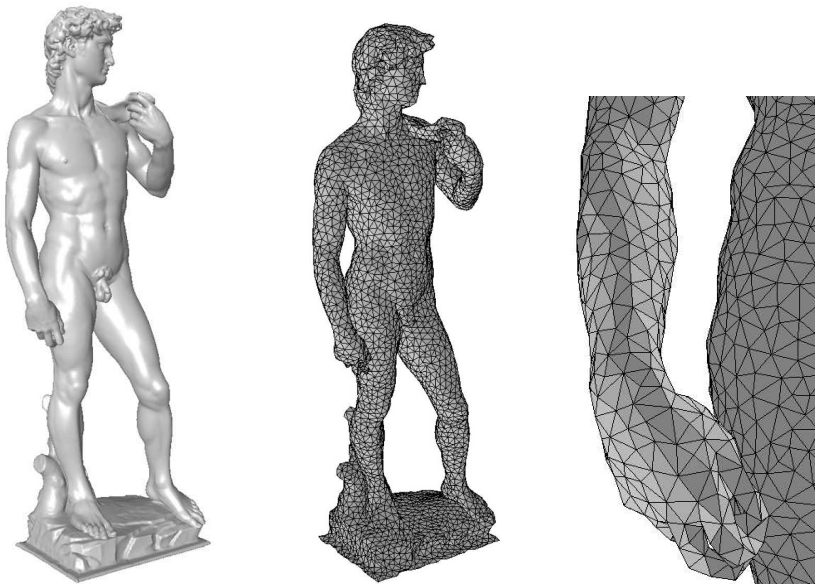


Fig. 4.12 *Geodesic remeshing with an increasing number of points.*

$W(\tilde{x})$ is able to modify the sampling.

The weight $W(\tilde{x})$ that modulates the metric of the 3D surface can be computed using a texture map. One can use a gradient-based metric as defined in (3.12), in order to put more samples in regions of large variation in the texture, see also Figure 3.12. Figure 4.15 shows an application of this idea to the adaptive remeshing of 3D faces.

We note that many alternative algorithms have been proposed for isotropic remeshing of surface according to a density field, see the review [5]. It is for instance possible to use a planar parameterization of the surface and use techniques from isotropic image sampling [8].



Original, 10^6 vertices Remeshed, 10^4 vertices Zoom

Fig. 4.13 *Uniform remeshing of the 3D David surface.*

4.4.2 Error-driven Surface Meshing

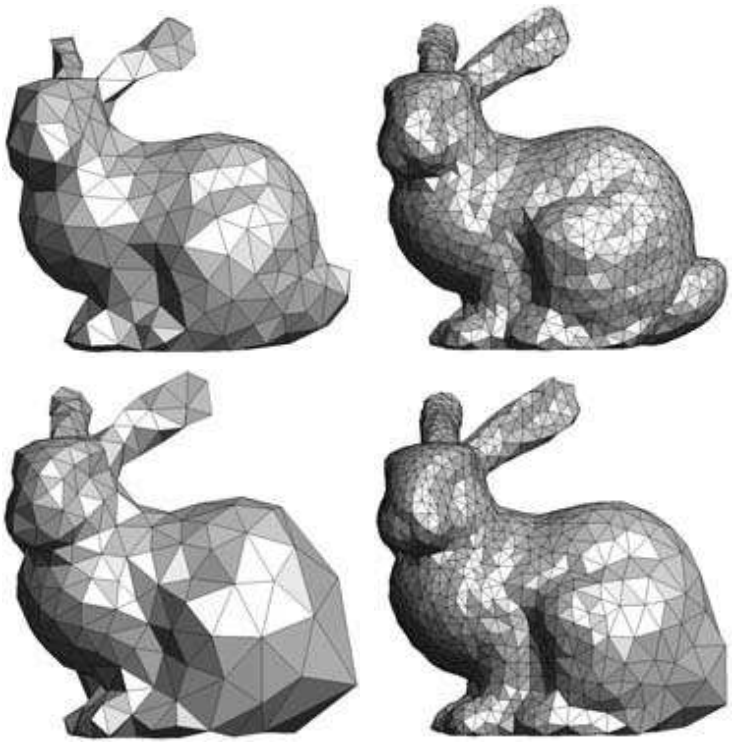
We denote by $\mathcal{S}_N$ the piecewise linear surface obtained from a triangulation $\mathcal{T}$. Instead of using a user defined metric $T_{\tilde{x}}$ for $\tilde{x} \in \mathcal{S}$, it is possible to design the metric to minimize the approximation error of $\mathcal{S}$ using $\mathcal{S}_N$. This problem extends the design of optimal triangulations to approximate images as exposed in Section 4.3.2. Indeed approximating an image $f(x_1, x_2)$ corresponds to the approximation of a parametric surface

$$(x_1, x_2) \in [0,1]^2 \mapsto (x_1, x_2, f(x_1, x_2)) \in \mathbb{R}^3. \tag{4.38}$$

Measuring distortion between surfaces is more difficult than measuring distances between functions as done in (4.18) and (4.19), because the set of surfaces is not a vector space, so that one cannot use classical functional norms.

The natural extension of the L^p distances to surfaces is the L^p Hausdorff distances

$$\delta_p(\mathcal{S}_1, \mathcal{S}_2) = \max(\tilde{\delta}_p(\mathcal{S}_1, \mathcal{S}_2), \tilde{\delta}_p(\mathcal{S}_2, \mathcal{S}_1)) \tag{4.39}$$



$N = 300$ $N = 1400$

Fig. 4.14 *Adaptive remeshing with a constant density (top) and a density linearly decreasing from left to right (bottom) .*

where the non-symmetric distance is

$$\tilde{\delta}_p(\mathcal{S}_1, \mathcal{S}_2)^p = \int_{\mathcal{S}_1} \min_{y \in \mathcal{S}_2} \|x - y\|^p \mathrm{d}x$$

and

$$\tilde{\delta}_\infty(\mathcal{S}_1, \mathcal{S}_2) = \max_{x \in \mathcal{S}_1} \min_{y \in \mathcal{S}_2} \|x - y\|.$$

Several algorithms perform fast approximate computations of these distances between meshes, see for instance [265, 65, 13], with applications to collision queries between surfaces [173].

Greedy schemes. Computing the optimized triangulated surface $\mathcal{S}_N$ to minimize $\delta(\mathcal{S}, \mathcal{S}_N)$ given some $N > 0$ is a difficult problem.

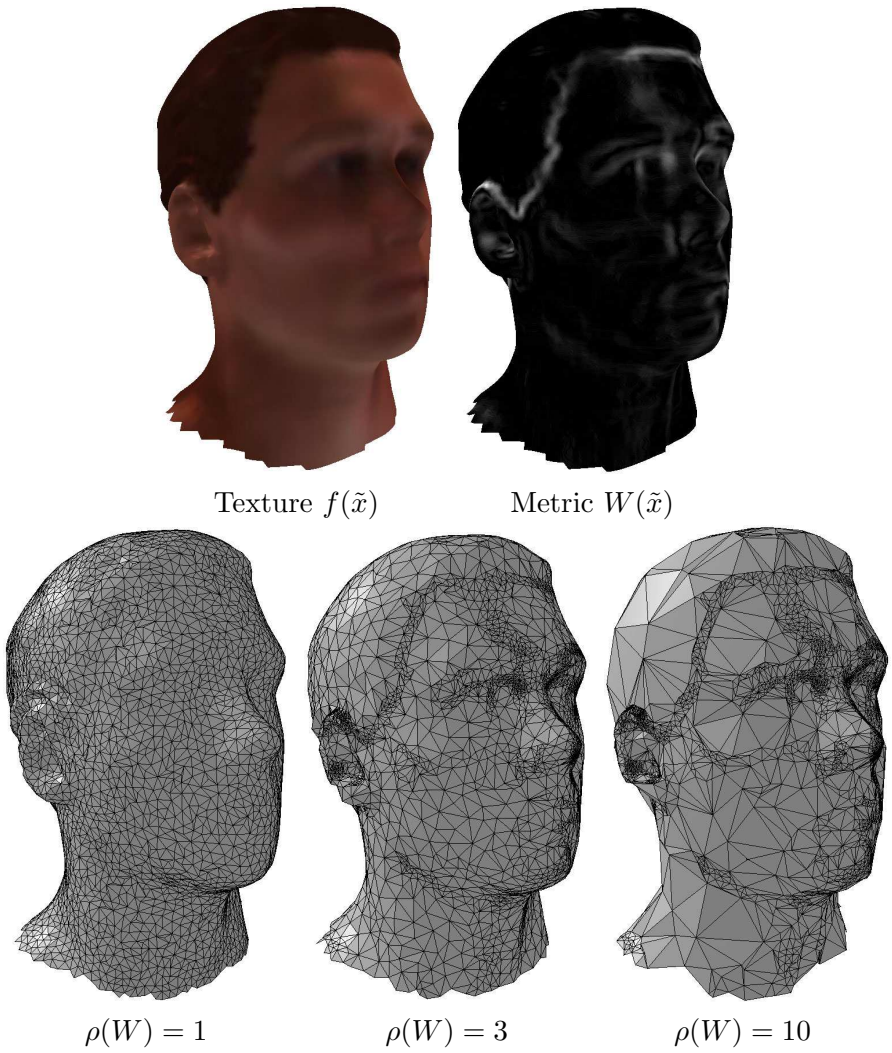


Fig. 4.15 *Adaptive remeshing with a density given by a texture. The adaptivity ratio* $\rho(W) = \max W / \min W$ *is increasing from left to right.*

One has to use sub-optimal greedy schemes that extend the methods detailed in Section 4.3.3 to the setting of surfaces. Popular algorithms include coarsening schemes that start from a dense triangulation of the surface [134, 122] and curve tracing methods that follow the curvature principal direction [6].

Since the Hausdorff metric (4.39) is significantly more difficult to compute and optimize than L^p norms, one has to use approximate metrics computed using various heuristics [134] or quadratic approximations [122].

Curvature-based isotropic metrics. Following the approach derived in Section 4.3.4 for functions, adapted approximations should follow the second order geometry of a smooth surface, and the Hessian matrix H_f defined in (4.21) is replaced by the second fundamental form J_φ defined in (3.14), where φ is a local parameterization. Indeed, for the special case of an altitude field (4.38), these two tensor fields are the same.

The extension of the isotropic metric (4.28) to surfaces makes use of the norm of the second fundamental form

$$W(\tilde{x})^2 = \|J_\varphi(x)\| \quad \text{where} \quad \tilde{x} = \varphi(x). \tag{4.40}$$

One can prove that an isotropic triangulation conforming to this metric leads to an asymptotic optimal approximation of C^2 surfaces for δ_∞. Note that this isotropic metric is optimized for approximation, and is in some sense the inverse of the metric (3.16) that is designed to force geodesic curves to follow salient features.

Figure 4.16, middle, shows such an example of curvature-adapted remeshing that improves the reconstruction of sharp features with respect to a uniform sampling, because more points are allocated in regions of high curvature.

Curvature-based anisotropic metrics. The quality of the approximation is further improved by making use of anisotropic triangulations. The extension of the anisotropic metric (4.33) to surfaces is the absolute value of the second fundamental form

$$T_{\tilde{x}} = |J_\varphi(x)| = |\mu_1(x)| e_1(x) e_1(x)^{\mathrm{T}} + |\mu_2(x)| e_2(x) e_2(x)^{\mathrm{T}}, \tag{4.41}$$

where the eigen-decomposition of the fundamental form is introduced in (3.15). One can prove that an anisotropic triangulation conforming to this metric leads to an asymptotic optimal approximation of C^2 surfaces for δ_∞ [67, 129, 128].

Figure 4.16 shows a comparison of surface remeshing using a constant metric, and metrics (4.40) and (4.41).

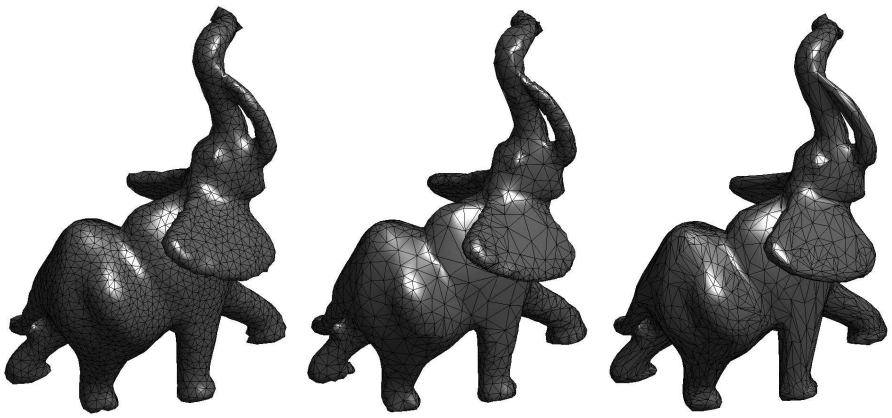


Fig. 4.16 *Comparison of constant, isotropic and anisotropic surface remeshing, with* $N = 3200$ *points.*

4.5 Domain Meshing

This section considers the meshing of a manifold with boundaries, which has important applications for numerical simulations with finite elements. We restrict ourselves to 2D manifolds with boundaries. Extension to higher dimensional manifolds makes use of the same line of ideas, but it is significantly more difficult to maintain mesh elements with good quality.

Section 4.5.1 presents a generalization of the farthest point sampling strategy, while Section 4.5.2 details the constraints that impose the meshing of the boundary of the manifold. Section 4.5.3 puts everything together and details the extension of the farthest point meshing to handle boundaries.

4.5.1 Delaunay refinement

The farthest point method automatically selects at each step the best point so that the sampling conforms to the Riemannian metric and is evenly spread over the domain according to the geodesic distance. It does not take into account the shape of the triangles, which is however important for some applications. For instance, for the numerical simulation of elliptic PDEs, it is necessary to have triangles that are as equilateral as possible. For other applications, triangles are allowed to have small angles, but should not have large angles, see [259]. It is possible to generalize the farthest point strategy using another selection rule among the set of local distance minimizers, which are the triple points defined in (4.7).

Triple point refinement. Except maybe during the first few iterations of the farthest point seeding, one notes that the farthest point selected by the algorithm is a triple point $x_{i,j,\ell}$ for $(i,j,k) \in \mathcal{T}(S)$, as defined in (4.7), or possibly a point located along the boundary. A generalization of this scheme inserts at each step an arbitrary triple point $x_{i,j,k}$ according to some quality measure $\rho(i,j,k)$. The greedy insertion rule (4.16) is replaced by

$$x_{k+1} = x_{i^\star,j^\star,k^\star} \quad \text{where} \quad (i^\star,j^\star,k^\star) = \operatorname*{argmax}_{(i,j,k)\in\mathcal{T}(S)} \rho(i,j,k). \tag{4.42}$$

The farthest point refinement corresponds to the quality measure

$$\rho(i,j,k) = d(x_i, x_{i,j,k}) = U_S(x_{i,j,k}). \tag{4.43}$$

In the Euclidean case $T_x = \mathrm{Id}_2$, one can prove that this generates uniform triangles with good quality, so that triangles do not have small angles [62].

A popular insertion rule, that also maintains triangles of good quality, but generates less triangles, selects a triangle $(i,j,k) \in \mathcal{T}(S)$ with the largest ratio of the circumradius to the shortest edge:

$$\rho(i,j,k) = \frac{d(x_{i,j,k}, x_i)}{\min(d(x_i,x_j), d(x_j,x_k), d(x_k,x_i))}. \tag{4.44}$$

This quantity can be computed for each triple point in parallel to the Fast Marching propagation.

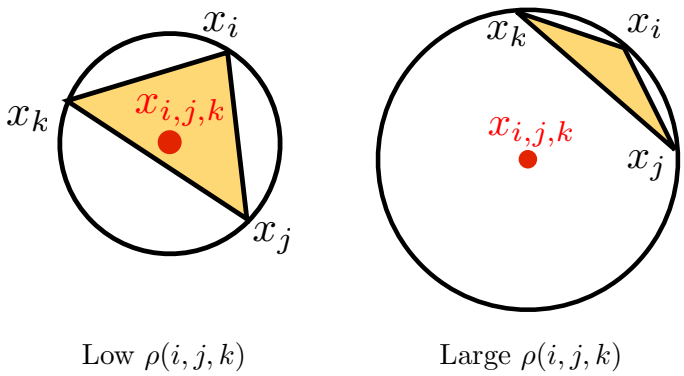


Fig. 4.17 *Examples of triangles with low (left) and large (right) aspect ratio.*

In the Euclidean domain, a triangle (x_i, x_j, x_k) with a large value of $\rho(i,j,k)$ is badly shaped since its smallest angle is close to 0, as shown in Figure 4.17. The selection rule (4.42) with the measure (4.44) thus tends to replace a badly shaped triangle by several smaller triangles of smaller size. It can be shown that similarly to the farthest point measure (4.43), the measure (4.44) leads to a triangulation without small angles [245, 61, 258].

As explained in [158], this property extends to an anisotropic metric T_x if angles are measured using the inner product defined by T_x. One should note that the measure (4.44) does not produce adapted triangulations that conform to the metric T_x since the length of the edges is not monitored.

Euclidean Delaunay refinement. In the Euclidean setting, these methods were introduced by Ruppert [245] and Chew [61, 62], see also [258] for an in depth analysis of these approaches, and [25] for a review of the methods. These methods choose at each iteration a triple point, which is a circumcenter of the Delaunay triangulation, while taking into account the boundary as explained in Section 4.5.2. These methods have been extended to build anisotropic meshes with a varying density using a local modification of the metric [34] or anisotropic elastic forces [36] and bubble packing [292].

4.5.2 Constrained Delaunay triangulation

For now we have either considered Riemannian manifolds without boundaries, or did not care about the reconstruction of the boundary. However, in some applications such as numerical simulation of PDEs, it is important that the meshing conforms to the boundary of the domain. In particular, the boundary of the discrete mesh should precisely approximate the boundary of the continuous manifold.

Manifold with boundary. In the following, we denote $i \leftrightarrow j$ to indicate that $x_i, x_j \in S \cap \partial\Omega$ are consecutive along the boundary (no other points $x_k \in \partial\Omega$ is between them).

To simplify the notations, we treat the outside of the shape as a Voronoi cell $\Omega^c = \mathcal{C}_\Xi$ associated to a virtual vertex x_Ξ, and consider the set of indices $I = \{\Xi, 0, \ldots, N-1\}$. This allows us to extend the notion of triple points (4.7) and Delaunay faces (4.6). This extension thus creates virtual exterior faces $(\Xi, i, j) \in \mathcal{T}(S)$ which indicates that two Voronoi cells $\mathcal{C}_i$ and $\mathcal{C}_j$ intersect at the boundary of the manifold. The associated triple point $x_{\Xi,i,j}$ thus lies along the boundary.

Constrained triangulation. To mesh correctly the boundary $\partial\Omega$ of the manifold, we require that it is part of the Delaunay graph $\mathcal{D}(S)$, which means that

$$\forall\, i \leftrightarrow j, \quad (i,j) \in \mathcal{D}(S).$$

This corresponds to a Delaunay triangulation constrained by the connections defined by the boundary.

This requirement is quite strong, since it might happen for an arbitrary geodesic Delaunay triangulation that a third point $x_k \in S$ encroaches the edge $i \leftrightarrow j$, which means that

$$(\Xi, i, k) \in \mathcal{T}(S) \quad \text{or} \quad (\Xi, j, k) \in \mathcal{T}(S).$$

In this situation $i \leftrightarrow j$ is not part of the Delaunay graph.

Figure 4.18, left, shows a valid situation where $i \leftrightarrow j$ is part of the Delaunay graph. Figure 4.18, right, shows a situation where x_k is close to the boundary $\partial\Omega$ and hence encroaches the edge $i \leftrightarrow j$.

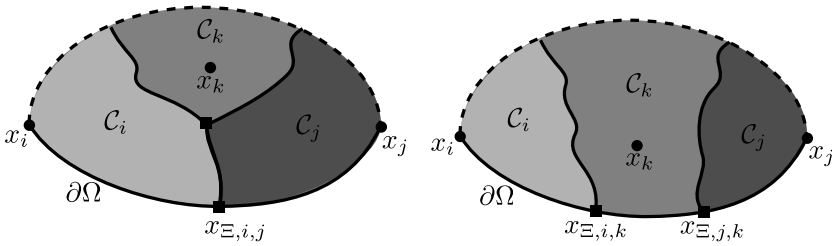


Fig. 4.18 *Left: the vertex x_k does not encroach the boundary curve $i \hookrightarrow j$ because (x_i, x_j) is a Delaunay edge. Right: the vertex x_k encroaches the boundary curve $i \hookrightarrow j$.*

4.5.3 Geodesic Delaunay refinement

It is possible to use the farthest point sampling algorithm, Algorithm 7, to mesh a manifold with boundary. Some care should be made in the algorithm so that the boundary of the domain is included in the Delaunay triangulation.

Encroaching vertex fixing. It might happen that a newly inserted farthest point x_k encroaches a boundary Delaunay edge $i \hookrightarrow j$. In this case, it is necessary to add to S a new boundary vertex $\tilde{x}_{i,j} \in \partial\Omega$ between x_i and x_j, that is selected at the same geodesic distance from the two boundary point

$$\tilde{x}_{i,j} \in \partial\Omega \quad \text{where} \quad d(x_i, \tilde{x}_{i,j}) = d(x_j, \tilde{x}_{i,j}). \tag{4.45}$$

Note that $\tilde{x}_{i,j}$ is not necessary equal to the double point $x_{i,j}$ defined in (4.5), since a double point is not constrained to lie on the boundary.

Isolated vertex fixing. As already noticed in Section 4.1.3, the Delaunay graph might not be a valid triangulation of the manifold. This is the case when a vertex x_i such that $(i, j) \in \mathcal{D}(S)$ is isolated, which means that it is not part of the triangulation

$$\forall\, k, \quad (i, j, k) \notin \mathcal{T}(S).$$

In this case, it is necessarily to add a new vertex $\bar{x}_{i,j} \in \Omega$ located on the Voronoi boundary between x_i and x_j, such as for instance

$$\bar{x}_{i,j} = \operatorname*{argmax}_{x \in C_i \cap C_j} d(x_i, x), \tag{4.46}$$

although other choices are possible.

Pseudo-geodesic Delaunay refinement. In order to incorporate global constraints within a provably correct Delaunay refinement scheme, Labelle and Shewchuk [158] make use of a Riemannian metric T_x and use the pseudo-distance

$$\tilde{d}(x,y)^2 = (x-y)^{\mathrm{T}} T_x (x-y). \tag{4.47}$$

Note that $\tilde{d}$ is not equal to the geodesic distance $d(x,y)$ unless T_x is constant. In particular it is not symmetric and does not satisfy the triangular inequality. For instance, Voronoi regions according to $\tilde{d}$ might have several connected components, which makes them more difficult to handle.

If one considers close enough points x, y, $\tilde{d}(x,y)$ is however a good approximation of the geodesic distance, and is easier to manipulate numerically. Labelle and Shewchuk [158] generalize Delaunay refinement using this pseudo-geodesic metric $\tilde{d}$, and they prove that for a large enough number of points, this algorithm produces a correct triangulation conforming to the metric field.

This algorithm is extended in 3D by [30, 31] and to domains with curves by [294]. This pseudo-geodesic distance $\tilde{d}$ has also been applied to image sampling [115] and surface remeshing [279].

Geodesic Delaunay refinement. It is possible to truly extend the Delaunay refinement to the manifold setting by generalizing the geodesic farthest point sampling and meshing [38]. This necessitates to compute geodesic distances on a fine grid using the numerical schemes detailed in Chapter 2, but creates a high quality mesh even if the number of samples is quite low, because the geodesic distance $d(x,y)$ integrates better the variations and the anisotropy of the metric T_x than the pseudo-distance $\tilde{d}(x,y)$ does.

A geodesic domain meshing algorithm is proposed in [38], which generalizes the approach of [158] by making use of the true geodesic distance inside the domain. It iteratively inserts the triple point $x_{i,j,k}$ with the largest aspect ratio $\rho(i,j,k)$. During the iterations, boundary middle points $\tilde{x}_{i,j}$ defined in (4.45) and isolation fixing points $\bar{x}_{i,j}$ defined in (4.46) are added. This maintains the geodesic Delaunay triangulation as a valid planar constrained triangulation of Ω.

A bound η_ρ on ρ enforces the refinement to reach some quality criterion, while a bound η_U on U_S enforces a uniform refinement to match some desired triangle density.

Algorithm 8 details this algorithm. Note that this algorithm only requires a local update of the distance map U_S and the Voronoi segmentation when a new point is added, so its complexity is similar to the complexity of the farthest point algorithm.

Similarly to the meshing method [158] with the pseudo geodesic distance (4.47), one can prove that this algorithm provides a valid triangulation of the domain if the metric does not have large variations.

4.5.4 Examples of Geodesic Domain Meshing

Isotropic Geodesic Refinement Examples. Figure 4.19, left, shows an example of uniform domain meshing using a constant metric $T_x = \mathrm{Id}_2$ together with this modified farthest point method.

It is possible to make use of an isotropic metric $T_x = W(x)^2 \mathrm{Id}_2$ to modulate the sampling density. One can define

$$W(x)^2 = \psi(d(x, \partial\Omega)),$$

where ψ is a decaying function and $d(x, \partial\Omega)$ is the distance to the boundary, which is the distance transform defined in Section 2.7. This metric tends to seed more points on the boundary of the shape Ω.

Another popular choice makes use of the local feature size, which is

Algorithm 8: Geodesic planar domain meshing algorithm.

Initialization: set S so that $\partial\Omega$ is covered by $\mathcal{D}(S)$.
repeat
  Boundary enforcement: while it exists $i \leftrightarrow j$ encroached, add $S \leftarrow S \cup \{\tilde{x}_{i,j}\}$.
  Triangulation enforcement: while it exists $(i,j) \in \mathcal{D}(S)$ with x_i isolated, add $S \leftarrow S \cup \{\bar{x}_{i,j}\}$.
  Select point: $(i^\star, j^\star, k^\star) = \underset{(i,j,k)\in\mathcal{T}(S)}{\operatorname{argmax}} \rho(i,j,k)$. Add it: $S \leftarrow S \cup \{x_{i^\star,j^\star,k^\star}\}$.
until $\rho(i^\star, j^\star, k^\star) < \eta_\rho$ *and* $U_S(x_{i^\star,j^\star,k^\star}) < \eta_U$;

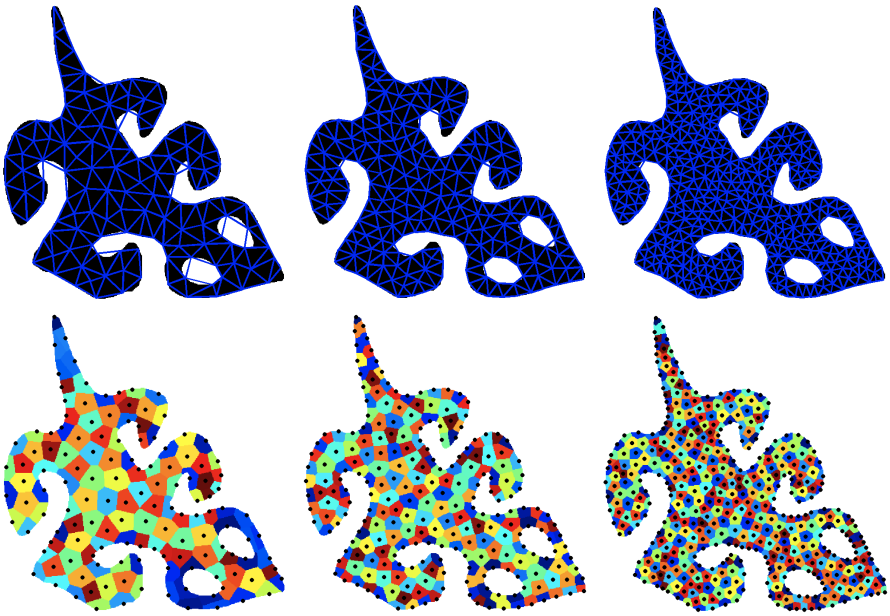


Fig. 4.19 *Uniform shape meshing with an increasing number of points, with* $T_x = \mathrm{Id}_2$.

the distance to the medial axis of the boundary

$$\forall\, x \in \Omega, \quad \gamma(x) = d(x, \mathrm{MedAxis}(\partial\Omega)).$$

The local feature size of the boundary is extended into the interior of the domain as a K-Lipschitz regular function

$$W(x)^{-1} = \min_{y \in \partial\Omega} K\|x - y\| + \gamma(y), \tag{4.48}$$

see for instance [7]. The rationale is that $W(x)$ is large in regions where the boundary has a large curvature, and inside thin elongated part of the shape Ω, where small equilateral triangles are required.

This K-Lipschitz sizing field $f(x) = W(x)^{-1}$ defined in (4.48) is the solution of an Eikonal equation

$$\forall\, x \in \Omega, \quad \|\nabla f(x)\| = K \quad \text{and} \quad \forall\, y \in \partial\Omega f(y) = \gamma(y).$$

Its solution can thus be approximated on a dense regular grid using the Fast Marching algorithm described in Section 2.3, using the value of $\gamma(y)$ as non-zero initial values on the boundary.

Example of Anisotropic Geodesic Refinement. Figure 4.20 shows an example of anisotropic meshing, obtained by the farthest point selection measure (4.43). The user controls the shape of the triangles by designing the tensor field T_x. In this example, the anisotropy of the metric is fixed, and the orientation of the tensor is defined by diffusing the orientation of the tangent to $\partial\Omega$ inside the domain.

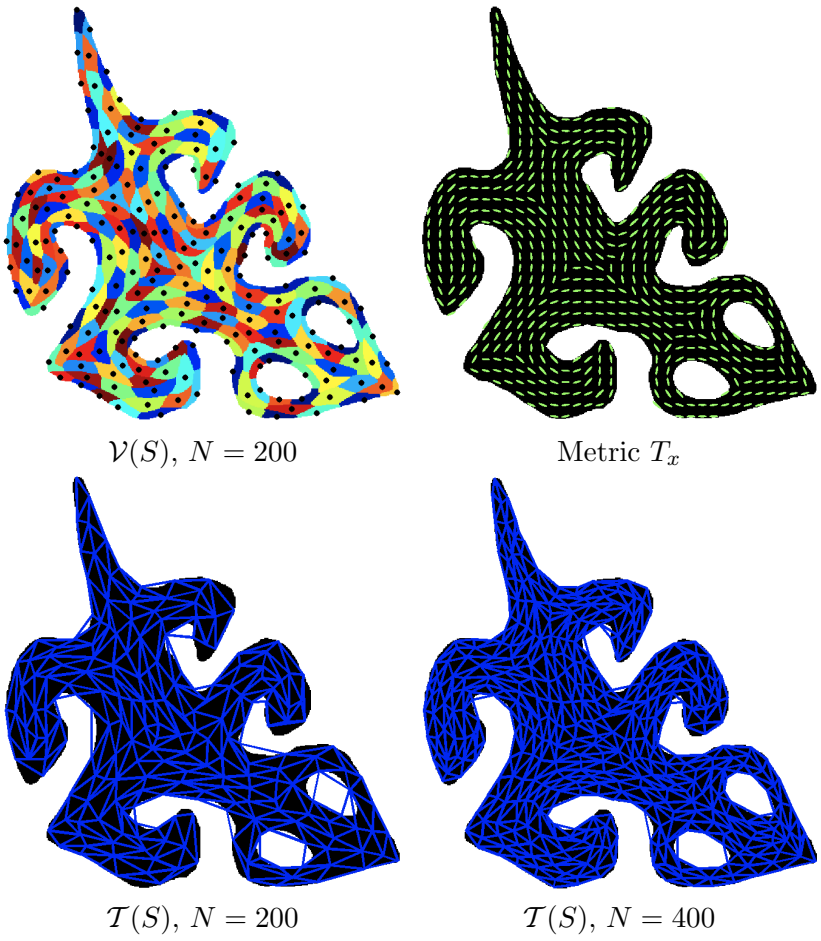


Fig. 4.20 *Example of anisotropic domain meshing.*

4.6 Centroidal Relaxation

Greedy sampling methods do not modify the location of an already seeded vertex. It is possible to enhance the quality of a greedy sampling by some global relaxation scheme, that moves points in S to minimize some energy $E(S)$. This energy depends on the problem the sampling is intended to solve. This section considers a quantization energy popular in applications ranging from clustering to numerical integration.

4.6.1 Optimal Quantization Problem

Given a budget N of points, an optimal sampling minimizes some energy

$$\min_{|S|=N} E(S). \tag{4.49}$$

A general rule to design an energy is to look for a sampling S that minimizes a weighted L^p norm over $\mathcal{M}$ of the geodesic distance map U_S to S

$$E(S) = \int_\Omega \rho(y) U_S(y)^p \mathrm{d}y = \int_\Omega \min_{i \in I} \rho(y) d(x_i, y)^p \mathrm{d}y, \tag{4.50}$$

where $\rho(y) > 0$ is a weighting function. Introducing the Voronoi segmentation $\mathcal{V}(S)$ defined in (4.1), this energy is re-written as

$$E(S) = \sum_{\mathcal{C}_i \in \mathcal{V}(S)} \int_{\mathcal{C}_i} \rho(y) d(x_i, y)^p \mathrm{d}y.$$

This minimization (4.49) corresponds to finding an optimal discrete sampling to approximate the continuous manifold Ω, and is referred to as an optimal quantization problem [129].

Optimal sets for the quadratic quantization cost ($p = 2$) can be shown to be asymptotically (when N is large) ε-nets, in the sense that they satisfy (4.9) and (4.10) for some ε that depends on N and the curvature of the manifold, see [67] and also [129, 128].

When the manifold is Euclidean $\Omega = \mathbb{R}^d$, the optimization of (4.50) becomes

$$\min_{|S|=N} \int_{\mathbb{R}^d} \rho(y) \min_{i \in I} \|x_i - y\|^p \mathrm{d}y. \tag{4.51}$$

This corresponds to the problem of vector quantization in coding [177]. This is also related to the problem of approximating a probability density ρ by a discrete density composed of Diracs at locations $\{x_i\}_{i\in I}$, and to the search for optimal cubature rules for numerical integration [107].

4.6.2 Lloyd Algorithm

The energy E is highly non-convex, and finding an optimal set of N points that minimizes E is difficult. One has to use an iterative scheme that converges to some local minimum of E. A good initialization is thus important for these schemes to be efficient, and one can for instance use an initial configuration computed with the farthest point algorithm detailed in Section 4.2.2.

Joint minimization. The minimization (4.49) on the points is replaced by a joint optimization on both the points and their associated regions

$$\min_{|S|=N} E(S) = \min_{|S|=N, \mathcal{V}\in\mathcal{P}_N(\Omega)} E(S,\mathcal{V}) = \sum_{\mathcal{C}_i\in\mathcal{V}} \int_{\mathcal{C}_i} \rho(y) d(x_i,y)^p \mathrm{d}y$$

where $\mathcal{P}_N(\Omega)$ is the set of partitions of the manifold Ω in N non-overlapping regions, so that $\mathcal{V}\in\mathcal{P}_N(\Omega)$ is equivalent to

$$\bigcup_{\mathcal{C}_i\in\mathcal{V}} \mathcal{C}_i = \Omega \quad \text{and} \quad \forall\, i\neq j,\ \mathcal{C}_i\cap\mathcal{C}_j = \partial\mathcal{C}_i \cap \partial\mathcal{C}_j.$$

Lloyd coordinate descent. Lloyd algorithm [177], originally designed to solve the Euclidean problem (4.51), minimizes alternatively $E(S,\mathcal{V})$ on the sampling point S and on the regions $\mathcal{V}$. It alternatively computes the Voronoi cells of the sampling, and then updates the sampling to be centroids of the cells. Algorithm 9 describes the Lloyd algorithm, and the next two paragraphs detail more precisely the two update steps that are iterated.

Region update. For a fixed sampling $S=\{x_i\}_{i\in I}$, one can see that the minimizer $\mathcal{V}^\star$ of $E(S,\mathcal{V})$ with respect to $\mathcal{V}$ is the Voronoi segmen-

tation

$$\mathcal{V}^\star = \operatorname*{argmin}_{\mathcal{V}\in\mathcal{P}_N(S)} E(S,\mathcal{V}) = \mathcal{V}(S). \tag{4.52}$$

Sample update. For a fixed tessellation $\mathcal{V} = \{\mathcal{C}_i\}_{i\in I} \in \mathcal{P}_N(\Omega)$ of the manifold, the minimizer of $E(S,\mathcal{V})$ with respect to S is

$$\operatorname*{argmin}_{|S|=N} E(S,\mathcal{V}) = \{c_p(\mathcal{C}_i)\}_{i\in I},$$

where the p-centroid $c_p(\mathcal{C})$ of a region $\mathcal{C}$ is

$$c_p(\mathcal{C}) = \operatorname*{argmin}_{x\in\Omega} E_{\mathcal{C}}(x) = \int_{\mathcal{C}} \rho(y) d(x,y)^p \mathrm{d}y. \tag{4.53}$$

If $p > 1$, this minimum is unique if $\mathcal{C}$ is small enough.

Convergence of the algorithm. The energy $E(S^{(\ell)})$ is decays with ℓ during the iterations of the Lloyd algorithm. One can show that this algorithm converges to some final sampling $S^\star$ under some restrictive hypothesis on the manifold [106]. The final sampling $S^\star$ is a local minimizer of the energy E, and is a so called centroidal Voronoi tessellation, because the samples are the centroids of the voronoi cells,

$$S^\star = \{c_p(\mathcal{C}_i^\star)\}_{i\in I} \quad \text{where} \quad \{\mathcal{C}_i^\star\}_i = \mathcal{V}(S^\star) \tag{4.54}$$

where the centroid is defined in (4.53). Centroidal Voronoi tessellations find many applications, see for instance the review paper [107].

The functional E can be shown to be piecewise smooth [176]. It is thus possible to use more efficient optimization methods to converge faster to a local minimum, see for instance [105, 176].

Algorithm 9: Lloyd algorithm.

Initialization: set $S^{(0)}$ at random, $\ell \leftarrow 0$.
repeat
  Region update: $\mathcal{V}^{(\ell+1)} = \mathcal{V}(S^{(\ell)})$.
  Sample update: $\forall\, i \in I,\ x_i^{(\ell+1)} = c_p(\mathcal{C}_i^{(\ell+1)})$.
  Set $\ell \leftarrow \ell + 1$.
until $\|S^{(\ell)} - S^{(\ell-1)}\|_\infty < \eta$;

Euclidean Lloyd. In the case of a Euclidean manifold $\Omega = \mathbb{R}^d$ with a Euclidean metric $T_x = \mathrm{Id}_d$, $d(x,y) = \|x-y\|$, the minimization (4.53) is

$$c_p(\mathcal{C}) = \operatorname*{argmin}_{x \in \mathbb{R}^d} E_{\mathcal{C}}(x) = \int_{\mathcal{C}} \rho(y)\|x-y\|^p \mathrm{d}y. \tag{4.55}$$

For $p=2$, it corresponds to the center of gravity (average)

$$c_2(\mathcal{C}) = m(\mathcal{C}) = \frac{1}{\int_{\mathcal{C}} \rho} \int_{\mathcal{C}} \rho(y) y \mathrm{d}y. \tag{4.56}$$

For $p>1$, the functional $E_{\mathcal{C}}$ to minimize is smooth, and $c_p(\mathcal{C})$ can thus be found by gradient or Newton descent.

Figure 4.21 shows an example of iterations of the Euclidean Lloyd algorithm for $p=2$. The computation is performed inside a square $\Omega \subset \mathbb{R}^2$, so that Voronoi cells are clipped to constrain them to lie inside Ω.

For $p \leqslant 1$, $E_{\mathcal{C}}$ is not smooth, and $c_p(\mathcal{C})$ can be approximated by re-weighted least squares

$$c^{(k+1)} = \operatorname*{argmin}_{x \in \mathbb{R}^d} \int_{\mathcal{C}} \rho^{(k+1)}(y)\|x-y\|^2 \mathrm{d}y = \frac{1}{\int_{\mathcal{C}} \rho^{(k+1)}} \int_{\mathcal{C}} \rho^{(k+1)}(y) y \mathrm{d}y,$$

where the weights at iteration k are

$$\rho^{(k+1)}(y) = \rho(y)\|c^{(k+1)} - y\|^{p-2}.$$

For $p=1$, $c_1(\mathcal{C})$ is a multi-dimensional median of the set $\mathcal{C}$, that extends to arbitrary dimension the 1D median.

Relation to clustering. This algorithm is related to the K-means clustering algorithm [138] to cluster a large set of points $\{y_j\}_{j \in J} \subset \mathbb{R}^d$. K-means restricts the computation to discrete points in Euclidean space, so that (4.51) is replaced by

$$E(S) = \sum_{j \in J} \min_{i \in I} \rho_j \|x_i - y_j\|^p.$$

Step (4.52) corresponds to the computation of a nearest neighbor for each point y_j

$$\forall\, j \in J, \quad k_j^{(\ell)} = \operatorname*{argmin}_{i \in I} \|x_i^{(\ell)} - y_j\|.$$

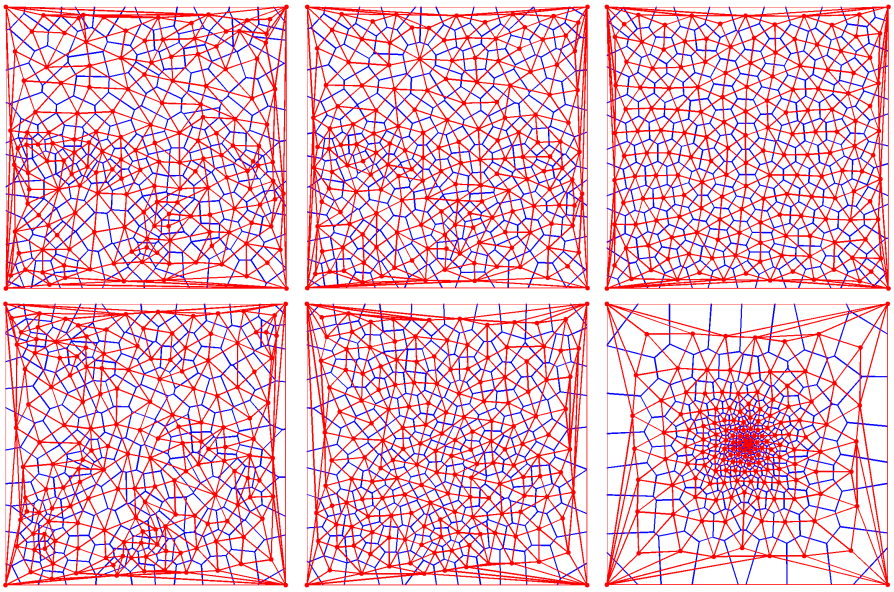


Fig. 4.21 *Iterations of Lloyd algorithm on a square with Euclidean metric, for a constant density $\rho = 1$ (top) and a varying density $\rho(x)$ larger for x in the middle of the square (bottom). Blue segments depict the Voronoi cells boundaries, and red segments represent the Delaunay triangulation.*

In the case $p = 2$, step (4.53) is replaced by an average

$$x_i^{(\ell)} = \frac{1}{\sum_{k_j^{(\ell)}=i} \rho_j} \sum_{k_j^{(\ell)}=i} \rho_j y_j.$$

4.6.3 Centroidal Tessellation on Manifolds

The update of the Voronoi cells (4.52) can be computed on arbitrary discrete manifolds as detailed in Section 2.6.1.

The computation of the centroid c_p in (4.53) is more difficult. When $p = 2$, it corresponds to an intrinsic center of gravity, also called Karcher or Frechet mean [142]. Such an intrinsic mean is popular in computer vision to perform mean and other statistical operations over high dimensional manifolds of shapes [154], see for instance [163].

Approximation by projection. If the manifold is embedded in a Euclidean space, so that $\Omega \subset \mathbb{R}^d$ for some d, it is possible to replace the geodesic distance $d(x,y)$ by the Euclidean one $\|x-y\|$ in $\mathbb{R}^d$ in (4.53), to obtain

$$\tilde{c}_p(\mathcal{C}) = \operatorname*{argmin}_{x\in\Omega} \tilde{E}_{\mathcal{C}}(x) = \int_{\mathcal{C}} \rho(y)\|x-y\|^p \mathrm{d}y$$

which is called a constrained centroid [109].

For the case $p=2$, it is shown in [109] that if $\tilde{c}_2(\mathcal{C})$ is a local minimizer of $\tilde{E}_{\mathcal{C}}$, then $\tilde{c}_2(\mathcal{C}) - m(\mathcal{C})$ is orthogonal to the surface at $\tilde{c}_2(\mathcal{C})$, where $m(\mathcal{C})$ is defined in (4.56).

One can thus compute a constrained centroid as the projection of the Euclidean center of mass

$$\tilde{c}_2(\mathcal{C}) = \mathrm{Proj}_{\mathcal{C}}(m(\mathcal{C})) \quad \text{where} \quad \mathrm{Proj}_{\mathcal{C}}(x) = \operatorname*{argmin}_{y\in\mathcal{C}} \|x-y\|.$$

If the Voronoi cells are small with respect to the curvature of the manifold, one can show that $\tilde{c}_2(\mathcal{C})$ is an accurate approximation of $c_2(\mathcal{C})$.

This constrained centroid method can be used to perform grid generation on surfaces, see [108].

Approximation by weighted centroid. For a Riemannian manifold over a parameterized domain $\Omega \subset \mathbb{R}^d$, it is possible to approximate the anisotropic metric T_x by an isotropic metric $T_x = W(x)^2 \mathrm{Id}_d$, for instance using the average of the eigenvalues of the tensors

$$W(x)^2 = \mathrm{Trace}(T_x)/d$$

and replacing the geodesic distance by a Euclidean one

$$d(x,y) \approx W(y)\|x-y\|,$$

which is accurate if x and y are close and if T_x is not too anisotropic. The original minimization (4.53) is then replaced by a weighted center of mass computation

$$\operatorname*{argmin}_{x\in\mathbb{R}^d} \int_{\mathcal{C}} \rho(y)W(y)\|x-y\|\mathrm{d}y = \frac{1}{\int_{\mathcal{C}} \rho W} \int_{\mathcal{C}} \rho(y)W(y)y\mathrm{d}y.$$

This method has been used for isotropic surface remeshing in [8]. In this setting, the manifold is 2-dimensional and corresponds to a 2D parameterization of a surface in $\mathbb{R}^3$.

Computation by gradient descent. When $p = 2$, if $\mathcal{C}$ is small enough, the function $E_{\mathcal{C}}$ is smooth, and its gradient can be computed as

$$\nabla E_{\mathcal{C}}(x) = \int_{\mathcal{C}} \rho(y) d(x,y) \frac{\gamma'_{x,y}(0)}{\|\gamma'_{x,y}(0)\|} \mathrm{d}y \tag{4.57}$$

where $\gamma_{x,y} \in \mathcal{P}(x,y)$ is the geodesic joining x and y such that $\gamma_{x,y}(0) = x$.

A local minimizer of $E_{\mathcal{C}}$ for $p = 2$ can be obtained by gradient descent, as proposed in [162, 291]. The computation of the gradient (4.57) is implemented numerically by performing a Fast Marching propagation starting from x, as detailed in Section 2.2.2, and then extracting geodesic curves $\gamma_{x,y}$ for discretized locations $y \in \mathcal{C}$ as detailed in Section 2.5.1.

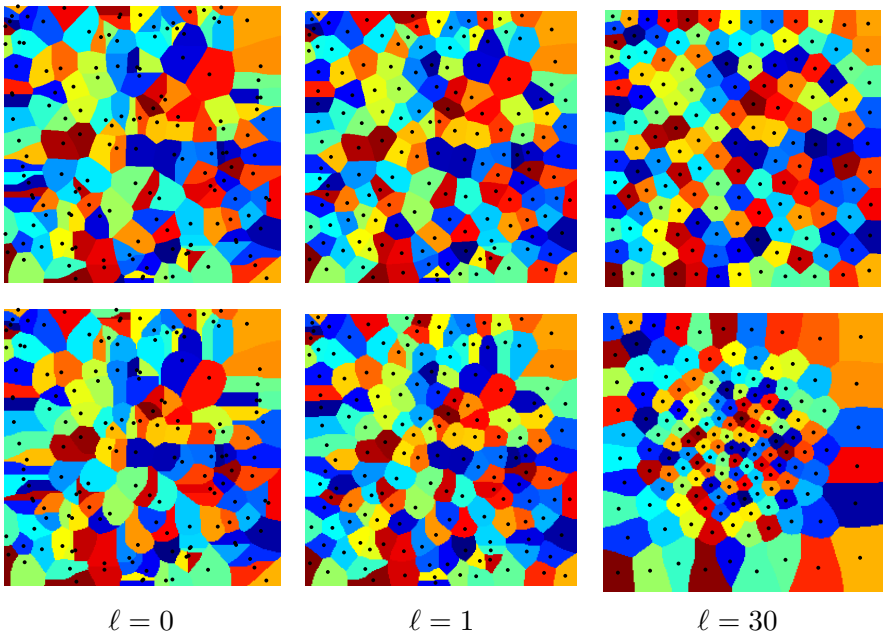


Fig. 4.22 *Iterations of Lloyd algorithm on a square with an isotropic metric* $T_x = W(x)^2 \mathrm{Id}_2$. *Top row: constant metric* $W(x) = 1$ *(Euclidean case), bottom row: varying metric* $W(x)$*, that is larger in the center.*

This method has been used in [219] to perform segmentation on 3D

surfaces. Figure 4.22 shows examples of the geodesic Lloyd algorithm on a square for an isotropic metric. Note that in this case, it gives results similar to the weighted Euclidean Lloyd, Figure 4.21, for a dense sampling. Figure 4.23 shows examples of iterations of the geodesic Lloyd algorithm on surfaces.

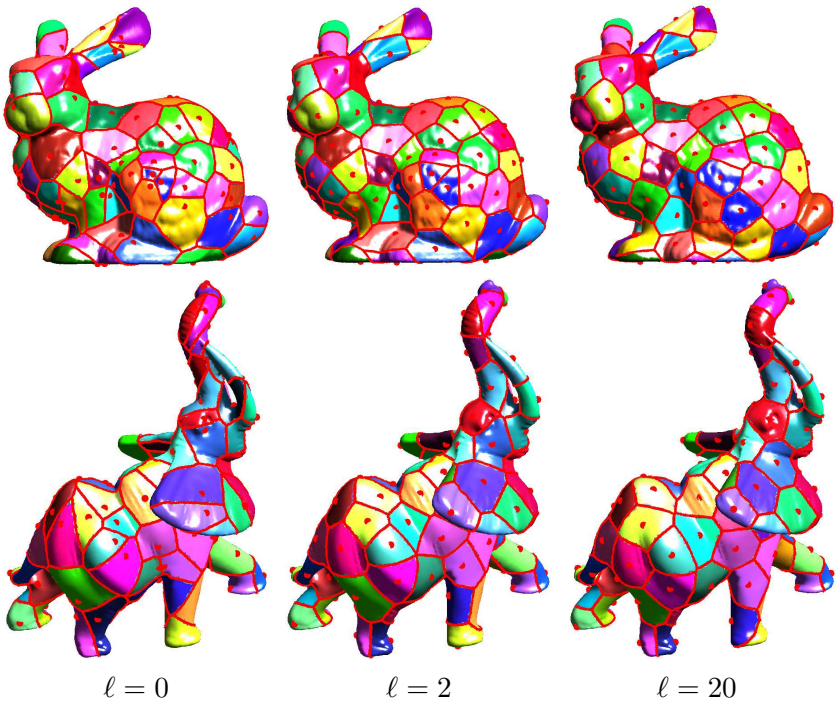


Fig. 4.23 *Iterations of Lloyd algorithm on surfaces.*

4.7 Perceptual Grouping

Perceptual grouping is a curve reconstruction problem where one wants to extract a curve from an image containing a sparse set of curves embedded in noise. This problem is relevant both to model good continuation perception laws [116, 289] and to develop efficient edge detection methods. In this paper, we restrict ourselves to the detection of a set of non-intersecting open or closed curves, although other kinds

of topological or regularity constraints could be enforced.

The idea of using anisotropic information to perform perceptual grouping was introduced in [130] where the completed contours are local minimizers of a saliency field. Many variational definitions of perceptual contours have been proposed using local regularity assumptions, for instance with the elastica model of Mumford [196], or good continuation principles [97].

Riemannian grouping. The grouping problem can be formulated as finding curves to join in a meaningful way a sparse set of points $S = \{x_i\}_{i \in I}$ while taking into account the information of a 2D image $f(x)$ for $x \in \Omega = [0,1]^2$. The regularity and anisotropy of f can be taken into account by designing a Riemannian metric T_x so that the set of curves are geodesics.

Cohen first proposed in [77] an isotropic metric $T_x = W(x)^2 \mathrm{Id}_2$, where $W(x)$ is a saliency field similar to those considered in Section 3.2 for active contours. This was extended to grouping of components in 2D and 3D images in [74, 73, 96]. This method was extended in Bougleux et al. [38] by designing a Riemannian metric T_x that propagates the anisotropy of the sparse curves to the whole domain. This anisotropic metric helps to disambiguate difficult situations where some curves are close to each other. This allows a better reconstruction with less user intervention.

The metric T_x is computed using the structure tensor as detailed in Section 4.3.5. The value of the structure tensor is retained only in areas where its anisotropy $A(x)$ defined in (1.17), is large, and the resulting tensor field is interpolated in the remaining part of the image, where no directional information is available. Figure 4.24 shows an example of anisotropic metric computed from an image representing a noisy curve.

This idea of interpolation of local orientation is similar to the computation of good continuation fields, as studied for instance in stochastic completion fields [289] or tensor voting [183].

Grouping by geodesic Delaunay pruning. The grouping algorithm proceeds by computing a perceptual graph $\tilde{\mathcal{D}}(S)$ of a set of points S provided by the user. This perceptual graph is a sub-graph of the

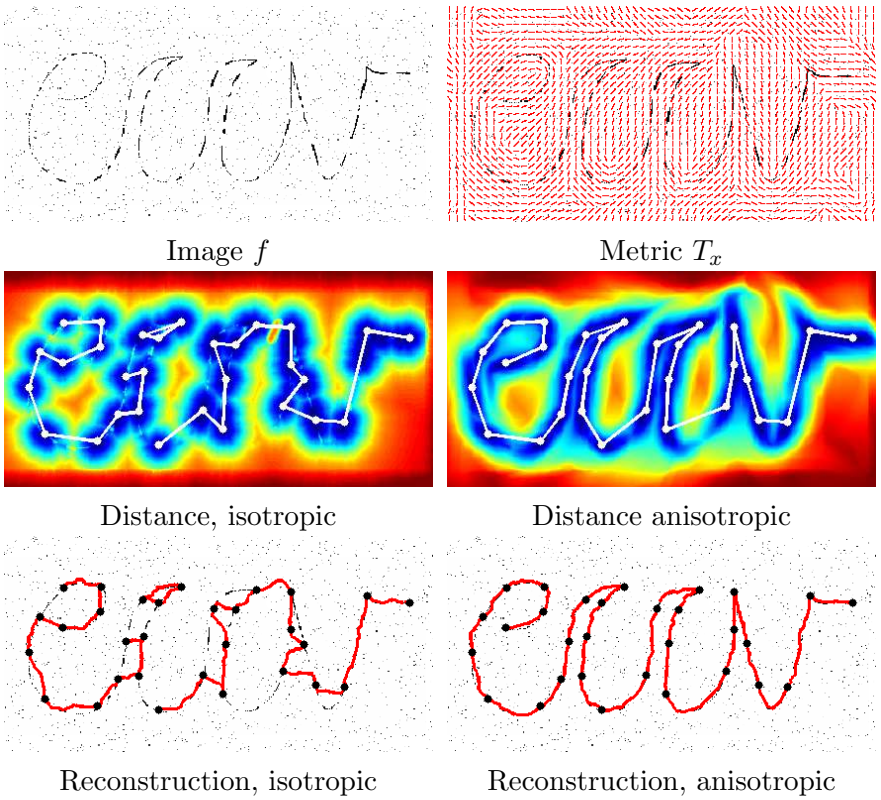


Fig. 4.24 *Peceptual grouping using isotropic metric (left) and anisotropic metric (right).*

Delaunay graph $\tilde{\mathcal{D}}(S) \subset \mathcal{D}(S)$. The graph is obtained by selecting in a greedy manner the shortest Delaunay edges. This algorithm is designed to extract curves without crossing, and the valence δ_i of vertices x_i in the perceptual graph is constrained to $\delta_i \leqslant 2$.

Algorithm 10 gives the details of the method. It is possible to extend this algorithm to add topological constraints on the curves to detect, or to allow several curves to meet at a crossing point.

This algorithm can be seen as a geodesic extension of methods for curve reconstruction that makes use of the Euclidean Delaunay graph [99]. Popular curve reconstruction methods [100, 9] connect points along combinatorial graphs derived from the Delaunay graph of the point set.

Figure 4.24 compares the results of perceptual grouping using an

isotropic metric as in [77] and using an anisotropic metric T_x as in [38]. The isotropic method fails because closed curves are connected regardless of their relative orientation. In contrast, the anisotropic metric enables a correct grouping of curves that obey a good continuation property.

Algorithm 10: Anisotropic perceptual grouping algorithm.

Initialization: $\tilde{\mathcal{D}}(S) \leftarrow \emptyset$, $\Pi \leftarrow \mathcal{D}(S)$, $\forall\, i \in I, \delta_i = 0$.
while $\Pi \neq \emptyset$ **do**
  Select edge: $(i^\star, j^\star) \longleftarrow \underset{(i,j)\in\Pi}{\operatorname{argmin}}\, d(x_i, x_j)$.
  Remove edge: $\Pi \leftarrow \Pi - \{(i^\star, j^\star)\}$.
  Check topology: if $\delta_i < 2$ *and* $\delta_j < 2$ **then**
    $\tilde{\mathcal{D}}(S) \leftarrow \tilde{\mathcal{D}}(S) \cup \{(x_i, x_j)\}$
    $\delta_j \leftarrow \delta_j + 1$ and $\delta_i \leftarrow \delta_j + 1$.

5

Geodesic Analysis of Shape and Surface

This chapter explores the use of geodesic distances to analyze the global structure of shapes and surfaces. This can be useful to perform dimensionality reduction by flattening the manifold on a flat Euclidean space, as detailed in Section 5.1. This flattening finds applications in mapping planar textures onto a surface, or in computing signatures that are invariant to non-rigid bendings. Correspondences between manifolds that respect the geodesic structure can be used to compare shapes and surfaces as shown in Section 5.2. To speed up applications of geodesic distances in shape retrieval, Section 5.3 designs compact histogram signatures.

The subject of non-rigid shape and surface matching, and in particular the use of geodesic distances, is exposed in much more details in the book [40].

5.1 Geodesic Dimensionality Reduction

Dimensionality reduction corresponds to mapping a manifold Ω of dimension d, possibly embedded in a high dimensional space $\Omega \subset \mathbb{R}^n$, $n > d$, into a Euclidean space $\mathbb{R}^k$ of small dimension $d \leqslant k < n$. This

reduction is performed using a mapping $\varphi : \Omega \to \mathbb{R}^k$, which is not necessarily bijective.

In the case where $d = k = 2, n = 3$, this allows to perform the flattening of a 3D surface onto a plane. In the case where $d = 2$ and $n = d$, this can also be used to replace a manifold Ω by a transformed one $\varphi(\Omega)$, so that $\varphi(\Omega) = \varphi(R\Omega)$ for a family $R \in \mathcal{R}$ of deformations. Using $\varphi(\Omega)$ instead of Ω for shape comparison leads to invariance to $\mathcal{R}$ of the method.

5.1.1 Multi-dimensional Scaling

To maintain geodesic distances during the mapping, one wishes to find $\varphi : \Omega \to \mathbb{R}^k$ so that

$$\forall\, x, y \in \Omega, \quad d_\Omega(x, y) \approx \|\varphi(x) - \varphi(y)\| \tag{5.1}$$

where d_Ω is the geodesic distance on manifold Ω, as defined in (1.15). Figure 5.1, left and center, shows examples of such a mapping that approximately maintains distances between pairs of points.

This problem is solved numerically by considering a set of points $\{\tilde{x}_i\}_{i=0}^{N-1} \subset \Omega$ discretizing the manifold, and by computing the position of $x_i = \varphi(\tilde{x}_i)$ such that

$$\forall\, 0 \leqslant i, j < N, \quad \|x_i - x_j\| \approx d_\Omega(\tilde{x}_i, \tilde{x}_j) = d_{i,j}.$$

This corresponds to the Multi-dimensional Scaling (MDS) problem [157, 32].

Projection on Euclidean Matrices. The positions $\{x_i\}_i$ are obtained by minimizing a given loss criterion. One can find the discrete mapping $x_i = \varphi(\tilde{x}_i)$ by computing

$$\min_{\{x_i\}_i} \sum_{0 \leqslant i,j < N} \delta(d_{i,j}, \|x_i - x_j\|), \tag{5.2}$$

where $\delta(a, b)$ is a given loss function. Each loss function δ leads to a different MDS method.

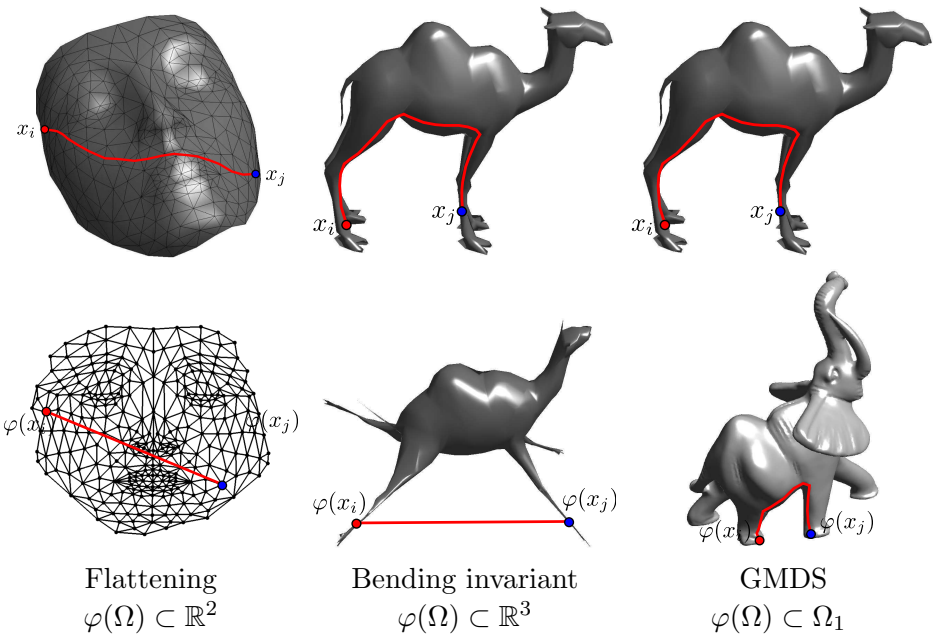


Fig. 5.1 *Overview of MDS and GMDS to compute embedding while conserving the pairwise distances. Top row: original manifold $\Omega \subset \mathbb{R}^3$. Bottom row: mapped manifold $\varphi(\Omega)$.*

Denoting as $X \in \mathbb{R}^{k \times N}$ the matrix whose columns are the positions $x_i \in \mathbb{R}^k$ of the points, one can rewrite this minimization as

$$X^\star \in \operatorname*{argmin}_{X \in \mathcal{C}_k} \delta(D(X), D) \quad \text{where} \quad \left\{ \begin{array}{l} D_{i,j} = d_{i,j}^2, \\ D(X)_{i,j} = \|x_i - x_j\|^2, \end{array} \right. \tag{5.3}$$

where δ is extended to matrices as

$$\delta(A, B) = \sum_{0 \leqslant i,j < N} \delta(A_{i,j}, B_{i,j}),$$

and where $\mathcal{C}_k$ is the set of centered points clouds

$$\mathcal{C}_k = \{X \setminus X\mathbb{I} = 0\}$$

where $\mathbb{I} \in \mathbb{R}^N$ is the constant vector of value 1.

The minimization (5.3) corresponds to the computation of the projection of the squared geodesic distance matrix D on the set $\mathcal{E}_k = \{D(X) \setminus X \in \mathcal{C}_k\}$ of squared k-dimensional Euclidean distance matrices, according to the distance δ between matrices.

The set $\mathcal{E}_k$ is non-convex, and a whole family of valid projections can be deduced from a single one. They are obtained from some $X^\star$ by applying rigid transformations in $\mathbb{R}^k$ (translation, rotation and symmetries). Indeed, applying such a transform to $X^\star$ does not change the distance matrix $D(X^\star)$.

Classical MDS approximation. For $\delta(a,b) = |a-b|^2$, the problem (5.3) corresponds to the Euclidean projection on the set of Euclidean distance matrices

$$X^\star \in \operatorname*{argmin}_{X \in \mathcal{C}_k} \|D(X) - D\|. \tag{5.4}$$

Unfortunately, the set $\mathcal{E}_k$ is non-convex, and computing the projection (5.4) is difficult.

In the following, we denote the centering operator as

$$J = \mathrm{Id}_N - \frac{1}{N}\mathbb{I}\mathbb{I}^{\mathrm{T}} \in \mathbb{R}^{N \times N}. \tag{5.5}$$

where $\mathbb{I} = (1, \ldots, 1)^{\mathrm{T}} \in \mathbb{R}^N$ is the constant vector, and $\mathbb{I}\mathbb{I}^{\mathrm{T}}$ is the constant matrix filled with ones. It maps $X \in \mathbb{R}^{k \times N}$ to a centered set of points $XJ \in \mathcal{C}_k$.

This centering operator allows to define another loss criterion called strain to replace the projection (5.4) by

$$\min_{X \in \mathcal{C}_k} \|J(D(X) - D)J\| \tag{5.6}$$

It turns out that one can find the global minimizer of the strain with an explicit formula, that we now detail. This corresponds to the so-called classical MDS [32].

Using the expansion

$$D(X) = d\mathbb{I}^{\mathrm{T}} + \mathbb{I}d - 2X^{\mathrm{T}}X \quad \text{where} \quad d = (\|x_i\|^2)_i \in \mathbb{R}^N,$$

and the fact that $J\mathbb{I} = 0$ and $X = XJ$ for $X \in \mathcal{C}_k$, one obtains

$$\forall\, X \in \mathcal{C}_k, \quad -\frac{1}{2}JD(X)J = X^{\mathrm{T}}X.$$

And thus (5.6) is re-written as

$$\min_{X \in \mathcal{C}_k} \|X^{\mathrm{T}}X + JDJ/2\|. \tag{5.7}$$

The solution $X^\star$ to this problem is in general unique (up to a rotation and symmetry of the points) and computed by first diagonalizing the symmetric matrix $K = -JDJ/2$

$$K = U^{\mathrm{T}}\Lambda U \quad \text{where} \quad \begin{cases} \Lambda = \mathrm{diag}(\lambda_0, \ldots, \lambda_{N-1}), \\ \lambda_i \geqslant \lambda_{i+1} \end{cases}$$

and then retaining only the first k leading eigenvectors

$$X^\star = \sqrt{\Lambda_k} U_k \quad \text{where} \quad \begin{cases} U_k = (u_0, \ldots, u_{k-1})^{\mathrm{T}}, \\ \Lambda_k = \mathrm{diag}(\lambda_0, \ldots, \lambda_{k-1}), \end{cases} \tag{5.8}$$

where u_i is an eigenvector of K, $Ku_i = \lambda_i u_i$.

The decay of eigenvalues λ_i indicates the dimensionality of the manifold. In particular, if $\lambda_i = 0$ for $i \geqslant k = 2$, it means that the (discrete) manifold Ω is isometric to the Euclidean plane, and that the classical MDS finds a correct embedding of the manifold.

Figure 5.3, middle column, shows example of mappings $\varphi : \Omega \to \mathbb{R}^k$ for $k = 2$ (top) and $k = 3$ (bottom) computed using classical MDS.

Local minimization using SMACOF. The approximation that replaces the matrix projection (5.3) by the strain minimization (5.6) is only used for computational simplicity, and does not come from a geometrical or physical motivation. In particular, the matrix J significantly changes the L^2 norm used to compute the projection, which might lead to a dimension reduction of poor quality.

To compare non-squared distances one defines the loss $\delta(a, b) = |\sqrt{a} - \sqrt{b}|^2$. Using (5.2) for this loss function, the dimensionality reduction is achieved by minimizing

$$\min_X S(X) = \sum_{0 \leqslant i,j < N} |\|x_i - x_j\| - d_{i,j}|^2. \tag{5.9}$$

The functional S in (5.9) is called the stress function. It is a smooth non-linear function at configurations such that $x_i \neq x_j$ for all $i \neq j$, which can be optimized using gradient descent to converge to a local minimum [157].

The SMACOF (scaling by majoring a complicated function) method [89] is a fast algorithm to solve this equation. It replaces the non-convex

minimization problem (5.9) by a series of simple convex problems. This algorithm computes iteratively $X^{(\ell)} \in \mathbb{R}^k \times N$ with a multiplicative update rule

$$X^{(\ell+1)} = X^{(\ell)} B(X^{(\ell)})/N \quad \text{where} \quad B(X)_{i,j} = \frac{d_{i,j}}{\|x_i - x_j\|}. \tag{5.10}$$

Note that this multiplicative iteration is equivalent to a gradient descent with a fixed step size [50]. One can prove that $X^{(\ell)}$ converges to a local minimum of the original one, see [89]. Since S is not convex, minimization with such an iterative method requires a good initialization $X^{(0)}$. This method can be accelerated using multi-grid computations [50] and vector extrapolation [239].

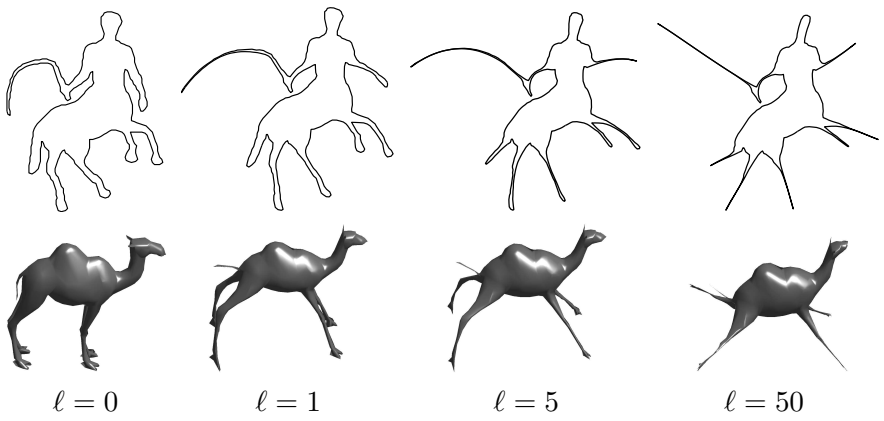


Fig. 5.2 *Iterations of the SMACOF algorithm.*

Figure 5.2, top row, shows the iteration of the SMACOF algorithm to compute a mapping $\varphi : \Omega \to \mathbb{R}^2$ where Ω is a 2D planar shape, as described in Section 1.3.2. Bottom row shows the SMACOF algorithm on a 3D surface Ω, to compute a mapping $\varphi : \Omega \to \mathbb{R}^3$.

Landmark acceleration. To speed up computation, one can use a set of Landmark points $\{x_i\}_{i=0}^{N_0-1} \subset \{x_i\}_{i=0}^{N-1}$ of the fully discretized manifold. One then only uses N_0 Fast Marching or Dijkstra propagations to compute the set of $N \times N_0$ distances

$$\forall\, 0 \leqslant i < N_0, \quad \forall\, 0 \leqslant j < N, \quad d_{i,j} = d_\Omega(x_i, x_j).$$

The goal is then to compute the embedding $\varphi(x_i)$ for all $0 \leqslant i < N$ from the partial distances $d_{i,j}$.

A first class of methods consists in applying any MDS method to find an embedding of the landmarks points alone, so that

$$\forall\, 0 \leqslant i,j < N_0, \quad \|\varphi(x_i) - \varphi(x_j)\| \approx \|x_i - x_j\|$$

T hen, one needs to interpolate the mapping φ to the remaining $N - N_0$ points, using various interpolation formula, see for instance [43]. For classical scaling, it is possible to use the eigenvector relationship, which corresponds to a Nistrom extrapolation, see [90].

For stress minimization, it is possible to minimize a partial stress

$$\min_{\{x_i\}_{i=0}^{N-1}} \sum_{i=0}^{N-1} \sum_{j=0}^{N_0-1} |d_{i,j} - \|x_i - x_j\||^2, \tag{5.11}$$

for which an extended SMACOF algorithm can be used, that extends the multiplicative update rule (5.10) and requires the resolution of a linear system at each iteration.

5.1.2 Bending Invariants

Invariant signatures. To perform shape and surface recognition in a way that is invariant to isometric deformations, one can replace a manifold Ω by its bending invariant $\varphi(\Omega)$ as defined in [111]. The mapping $\varphi : \Omega \to \mathbb{R}^k$ is computed by solving the MDS problem, using either the classical scaling solution (5.8) or the stress minimization (5.9).

The bending invariant is originally designed to replace a 3D surface $\Omega \subset \mathbb{R}^3$ by a signature $\varphi(\Omega) \subset \mathbb{R}^3$ in the same embedding space. Figure 5.3, top row, shows an example of such an invariant signature.

It can also be applied to a binary shape $\Omega \subset \mathbb{R}^2$ that is a compact planar set, as described in Section 1.3.2. In this case, $\varphi(\Omega) \subset \mathbb{R}^2$ is a deformed shape, as shown on Figure 5.3, bottom row.

Extrinsic comparison of signatures. Given two manifolds Ω_0 and Ω_1, one can compare shapes up to isometric deformations by comparing their bending invariant $\varphi_i(\Omega_i)$. In this way it is possible to define

a distance between manifolds that is invariant to isometries such as bendings or articulations

$$\Delta(\Omega_0, \Omega_1) = \min_{R \in \mathcal{R}(\mathbb{R}^k)} \delta(\varphi_0(\Omega_0), R(\varphi_1(\Omega_1))), \tag{5.12}$$

where $\mathcal{R}(\mathbb{R}^k)$ is the set of Euclidean isometries of $\mathbb{R}^k$ (rotations, translations and symmetries), and δ is a distortion measure between subsets of $\mathbb{R}^k$, such as for instance the Hausdorff distance, defined in (5.13)

$$\delta(A_1, A_2) = \max(\tilde{\delta}(A_1, A_2), \tilde{\delta}(A_2, A_1)), \tag{5.13}$$

where the non-symmetric distance is

$$\tilde{\delta}(A_1, A_2) = \max_{x_1 \in A_1} \min_{x_2 \in A_2} \|x_1 - x_2\|.$$

Computing exactly Δ is difficult, but one can resort to approximate iterative algorithms such as iterative closest points [60, 27].

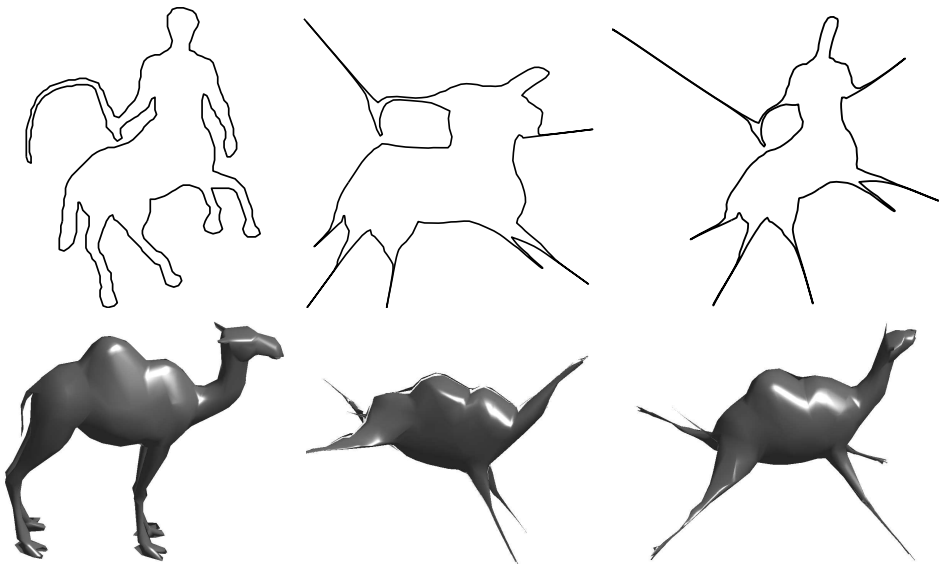


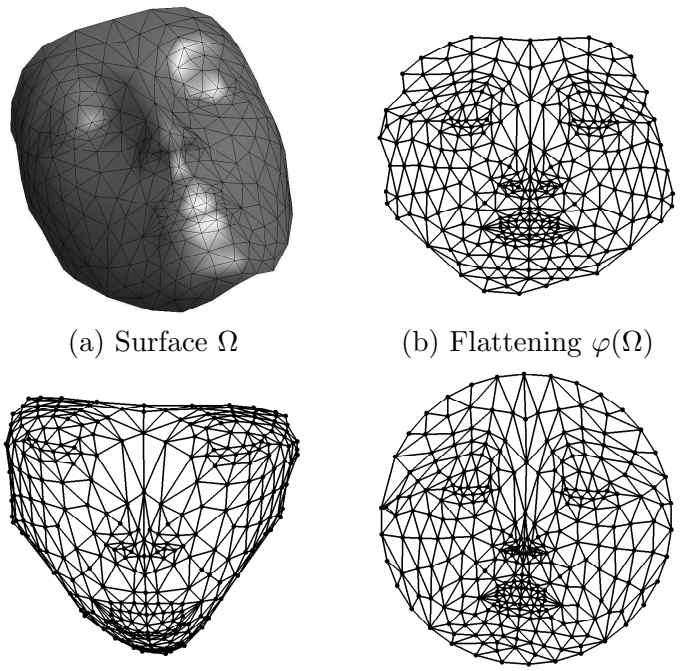
Fig. 5.3 *Examples of bending invariance computed using classical MDS and SMACOF, 2D shape (top row) and for a 3D surface (bottom row).*

5.1.3 Surface Flattening and Parameterization

Geodesic surface flattening. Surface flattening computes a map $\varphi : \Omega \to \mathbb{R}^2$ where Ω is a 2D manifold, typically embedded into 3D space $\mathbb{R}^n = \mathbb{R}^3$. This corresponds to a particular case of dimensionality reduction. One can thus use the MDS methods to compute such a flattening that maps the geodesic distances on Ω to Euclidean distances.

This approach was originally proposed by [250] to perform the flattening of the cortical surface. It was also applied in [302] to perform texture mapping.

Figure 5.4, (b), shows an example of such a geodesic surface flattening.



(a) Surface Ω (b) Flattening $\varphi(\Omega)$

(c) Laplacian eigenmaps (d) Laplacian parameterization

Fig. 5.4 *Examples of surface flattening and parameterization using spectral methods (b,c) and linear system resolution (d).*

Differential surface flattening. Many other approaches for surface flattening have been proposed, for instance by minimizing the deviation from conformality or area conservation. This leads to the computation of the mapping φ using eigen-vectors of certain differential operators, such as the Laplace-Beltrami second order derivative. This can be computed numerically, as detailed in the Laplacian eigen-maps framework [19].

Figure 5.4, (c), shows an example of such a Laplacian eigen-maps flattening, where the X and Y coordinates of the embedding are the second and third eigenvectors of a discretized Laplacian. Various discretizations exist, with the most popular being the cotangent weights, see for instance [257, 117] and references therein.

Differential surface parameterization. Other approaches flatten a 2D manifold Ω with the topology of a disk by imposing that the boundary $\partial\Omega$ is mapped onto a closed convex curve in $\mathbb{R}^2$. The minimization of differential distortion such as the deviation from conformality leads to a mapping φ that can be shown to be bijective [277]. This is useful to parameterize a surface for texture mapping application.

The X and Y coordinates of the embedding $\psi(\Omega)$ of a discretized triangulated surface Ω are then both solution of a linear system whose left hand size is a discrete Laplacian, and right hand side incorporates the fixed location of the boundary. These two systems can be efficiently solved using a sparse linear solver. See [257, 117] for surveys about mesh parameterization methods.

Figure 5.4, (d), shows an example of such a Laplacian parameterization, where the boundary of Ω is mapped by φ on a circle.

5.1.4 Manifold Learning

The application of MDS to geodesic distances is used in the Isomap algorithm [270] to perform manifold learning. In this setting, the manifold is estimated from a point cloud $\{x_i\}_i \subset \mathbb{R}^n$ using a nearest neighbor graph, and the geodesic distances are estimated using the Dijkstra algorithm detailed in Section 2.2.3.

The graph adjacency relationship can be defined in different ways.

The simpler one is obtained by thresholding the Euclidean distance in $\mathbb{R}^n$, and the graph metric is defined as

$$W_{i,j} = \begin{cases} \|x_i - x_j\| & \text{if} \quad \|x_i - x_j\| \leqslant \varepsilon, \\ +\infty & \text{otherwise.} \end{cases}$$

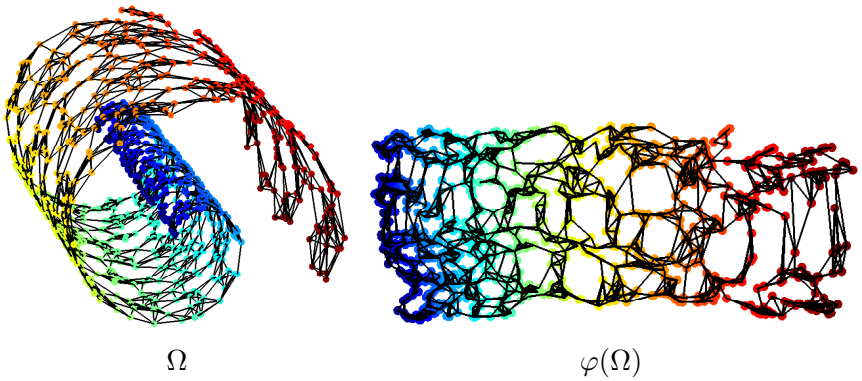


Fig. 5.5 *Example of dimensionality reduction using Isomap with classical MDS.*

Figure 5.5 shows an example of application of Isomap to a simple 3D point cloud that is sampled on a surface isometric to a planar rectangle. Figure 5.6 shows iterations of the SMACOF algorithm (5.10) on the same dataset. To overcome the numerical complexity of computing all pairwise distances, local spectral methods that enforce a local smoothness of the mapping φ have also been proposed, based on local tangent plane estimation [243], Laplacian [19] or Hessian operator [102]. These methods suffers from difficulties to handle a manifold with a complicated topology, but it is possible to enforce topological constraints during the learning [240].

Manifold learning can be used to recover the low dimensional geometry of a database of images, such as for instance binary digits or images representing an object under varying lighting and camera view. In practice though, this method works for relatively simple manifold with a simple topology, see [103] for a theoretical analysis of the geodesic geometry of image datasets.

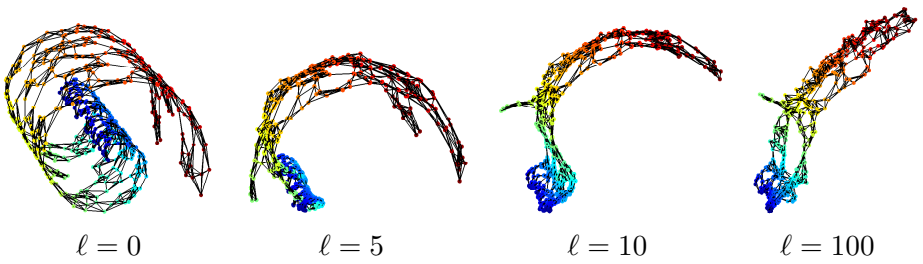


Fig. 5.6 *Iterations of SMACOF algorithm to compute the Isomap embedding.*

5.2 Geodesic Shape and Surface Correspondence

Shape and surface comparison can be performed by matching two manifolds Ω_0 and Ω_1. This can be achieved by mapping one manifold onto the other.

A correspondence map is a function between the manifolds

$$\varphi : \Omega_0 \longrightarrow \Omega_1.$$

Depending on the application, one usually restricts the set of allowable φ, such as for instance mappings respecting some invariance. By computing an optimal map that minimizes some distortion, one gains access to a measure of similarity between manifolds to perform retrieval (the distortion) and an explicit mapping (the optimal map itself) that can be used for several applications such as texture mapping.

Finding a mapping φ between two manifolds generalizes the dimensionality reduction problem (5.1) to the case where the second manifold is not Euclidean.

5.2.1 Mapping Between Manifolds and Distortion

The bending invariant distance (5.12) requires the use of an intermediate Euclidean space $\mathbb{R}^k$ to measure the similarity between two different manifolds Ω_0 and Ω_1. This approach can be simplified by mapping directly one manifold onto the other, in a way that preserves the geodesic distances.

In the following, d_{Ω_i} is the geodesic distance on the manifold defined in (1.15).

Similarities between manifolds. A measure of similarity $\Delta(\Omega_0, \Omega_1)$ between two manifolds takes into account some important features of the surface, and on the contrary discards other meaningless features to gain invariance in shape recognition.

One usually would like this measure to be a valid metric among the space of shapes, and in particular it should be symmetric and satisfy the triangular inequality

$$\Delta(\Omega_0, \Omega_1) \leqslant \Delta(\Omega_0, \Omega) + \Delta(\Omega, \Omega_1). \tag{5.14}$$

This desirable property implies that if both Ω_i are approximated by discretized manifolds at precision ε, meaning $\Delta(\Omega_i, \Omega_i^\varepsilon) \leqslant \varepsilon$, then the discrete measure of similarity is within precision 2ε from the true one

$$|\Delta(\Omega_0, \Omega_1) - \Delta(\Omega_0^\varepsilon, \Omega_1^\varepsilon)| \leqslant 2\varepsilon.$$

The condition (5.14) is however not satisfied by many useful similarity measures.

Correspondence and similarities. An optimal correspondence is selected by minimizing some distortion criteria $\delta(\varphi)$ in a restricted class of mapping. For some applications, one also wishes φ to be injective so that it does not map two different points of Ω_0 to the same location.

The measure of similarity Δ between manifolds is then computed as the distortion of this optimal map

$$\Delta(\Omega_0, \Omega_1) = \min_{\varphi:\Omega_0 \to \Omega_1} \delta(\varphi).$$

One should note that this similarity measure is non symmetric, and does not in general satisfy the triangular inequality (5.14).

5.2.2 As Isometric as Possible Correspondence.

To compare shapes according to their intrinsic geometry, one wishes to use a correspondence that maintains as much as possible the geodesic distance along the manifold.

Gromov-Hausdorf framework. The Gromov-Hausdorff measure [127, 51] particularized to the case of Riemannian manifolds, is

a metric between metric spaces that measures the joint distortion of pairs of mappings $(\varphi_{0,1}, \varphi_{1,0})$ between two manifolds

$$\varphi_{i,j} : \Omega_i \to \Omega_j.$$

The distortion of the pair $(\varphi_{0,1}, \varphi_{1,0})$ is measured using the maximum distortion of each map

$$\delta(\varphi_{i,j}) = \max_{x,y \in \Omega_i} |d_{\Omega_i}(x,y) - d_{\Omega_j}(\varphi_{i,j}(x), \varphi_{i,j}(y))|,$$

and a joint distortion

$$\delta(\varphi_{0,1}, \varphi_{1,0}) = \max_{x \in \Omega_0, y \in \Omega_1} |d_{\Omega_0}(x, \varphi_{1,0}(y)) - d_{\Omega_1}(\varphi_{0,1}(x), y)|.$$

The Gromov-Hausdorff distance between the two manifolds is then defined as

$$\Delta(\Omega_0, \Omega_1) = \min_{\varphi_0 : \Omega_0 \to \Omega_1, \varphi_1 : \Omega_1 \to \Omega_0} \max\left(\delta(\varphi_0), \delta(\varphi_1), \delta(\varphi_0, \varphi_1)\right). \quad (5.15)$$

One can show that this similarity measure Δ is a metric among manifolds, and in particular it satisfies the triangular inequality (5.14).

This Gromov-Hausdorff distance was introduced in computer vision by Memoli and Sapiro [187]. For discretized spaces $\Omega_0 = \{x_i\}_{i=0}^{N-1}$ and $\Omega_1 = \{y_i\}_{i=0}^{N-1}$, where we have used the same number N of points, and if one restricts its attention to bijective mappings $\varphi : \Omega_0 \to \Omega_1$, one can approximate the Gromov-Hausdorff distance (5.15) by a permutation distance

$$\Delta(\Omega_0, \Omega_1) = \min_{\sigma \in \Sigma_N} \max_{0 \leqslant i,j < N} |d_{\Omega_0}(x_i, x_j) - d_{\Omega_1}(x_{\sigma_i}, x_{\sigma_j})|, \quad (5.16)$$

where Σ_N is the set of permutation of N numbers. This distance can be shown to be a faithful approximation of (5.15) for randomized sampling, see [187].

Computing the distance (5.16) is computationally prohibitive, since it requires to check all possible permutations. A fast approximate algorithm was developed in [187] and has been applied to comparison and retrieval [185]. The minimization (5.16)can be recasted as a binary graph labeling problem [272], which is NP-hard in the general case, and can be approximated using fast algorithms [285]. The Gromov-Hausdorff distance has been relaxed to a probabilistic setting [186, 185],

where each manifold point is associated to a probability to take into account for imperfect measurements and partial matching. This defines a family of Gromov-Wasserstein distances that can be computed numerically by solving a non-convex optimization problem.

GMDS framework. The maximum error in the Hausdorff distance (5.15) is difficult to manipulate and optimize numerically. One usually prefers an average notion of geodesic deviation, such as for instance a mean square distortion

$$\delta(\varphi) = \iint_{\Omega_0^2} |d_{\Omega_0}(x,y) - d_{\Omega_1}(\varphi(x),\varphi(y))|^2 \mathrm{d}x\mathrm{d}y, \tag{5.17}$$

introduced by Bronstein, Bronstein and Kimmel in the Generalized Multi-dimensional Scaling (GMDS) shape matching framework [49]. Figure 5.1, right, shows a schematic example of this approximate conservation of pairwise geodesic distances between two surfaces.

In this setting, the integration measure $\mathrm{d}x$ refers to the area volume element $|\det(T_x)|$ defined on the manifold Ω_i from the Riemannian metric. In the usual case where the manifolds are embedded in $\mathbb{R}^d$, it corresponds to the usual area element in $\mathbb{R}^d$.

This mapping distortion (5.17) defines a non-symmetric distortion on the set of manifolds

$$\Delta(\Omega_0,\Omega_1) = \min_{\varphi:\Omega_0\to\Omega_1} \delta(\varphi).$$

One should be careful that, on the contrary to the Gromov-Hausdorff distance (5.15), Δ is not symmetric and does not satisfy the triangular inequality (5.14). It does not define a distance among manifolds.

The distortion (5.17) is computed numerically on a discretized manifold $\{x_i\}_{i=0}^{N-1}$ and the set of points $y_i = \varphi(x_i) \in \Omega_1$ that minimizes

$$\Delta(\Omega_0,\Omega_1) = \min_{\{y_i\}_i} \delta(\{y_i\}_i) = \min_{\{y_i\}_i} \sum_{0\leqslant i,j<N} |d_{\Omega_0}(x_i,x_j) - d_{\Omega_1}(y_i,y_j)|^2$$

The GMDS algorithm [49] finds a local minima of this complicated non-convex energy by gradient descent. It requires a proper interpolation of the geodesic distance d_{Ω_1} on Ω_1 that is usually pre-computed on

a discrete set of points, whereas the optimized location $\{y_i\}_i$ varies continuously.

The distance $\Delta(\Omega_0, \Omega_1)$ can be applied to surface [49] and shape retrieval [41] using nearest neighbors or more advanced classifiers. The partial stress (5.11)can be extended to the GMDS framework to take into account embedding of a manifold into a subset of another manifold. GMDS can be further extended to allow for partial matching [42]. GMDS has been extended to take into account photometric information [271].

The optimal mapping $\varphi^\star$ computed with GMDS, which is a local minima of $\delta(\varphi)$, is also relevant to perform shape comparison and processing. A 3D facial surface Ω_0 is embedded as $\varphi^\star(\Omega_0)$ into a sphere Ω_1, with minimal distortion by finding the optimal sphere radius. The resulting nearly isometric signature $\varphi^\star(\Omega_0)$ can then be used to perform 3D face recognition [45]. This optimal mapping $\varphi^\star$ can also be used to perform texture mapping of animated surfaces [44].

5.2.3 2D Shape Matching

Matching 2D shapes represented as closed contours is simpler than matching higher dimensional manifold. In particular, it can be solved with fast algorithms. The analysis and retrieval of 2D closed curves has received considerable attention, both because it is computationally tractable, and because of its relevance for human vision [195], which is highly sensitive to contours in images.

The structure of non-intersecting curves. In this setting, one considers a 2D planar shape Ω, and focuses on its contour $\partial\Omega$, which is assumed to be a closed non-intersecting curve $\gamma : [0,1] \to [0,1]^2$. This curve is closed and 1-periodic, meaning $\gamma(0) = \gamma(1)$.

The set of non-intersecting closed curves has a very special structure [163], that can be represented as an infinite-dimensional Riemannian space [163]. The resulting space of curves can be manipulated to define various operations such as shape averaging [217].

Finding a correspondence between shapes can be thought as finding a path connecting two curve on this high dimensional space of curves.

Taking into account this structure is however too complicated, and in practice one sets up ad hoc optimization problems that require finding shortest paths, as we detail next.

Matching between curves. We consider two shapes Ω_0, Ω_1, represented using their contour curves γ_0, γ_1. A bijective matching between two curves γ_0, γ_1 can be represented as a bijective map φ between the parameter spaces. Since these parameter domains are periodic, this corresponds to find some $\eta \in [0,1]$ and a non-decreasing map

$$\varphi : [0,1] \to [\eta, 1+\eta],$$

such that the local geometry around $\gamma_0(t) \in \Omega_0$ matches in some sense the local geometry around $\gamma_1(\varphi(t)) \in \Omega_1$.

Local differential features. For each point $x \in \partial\Omega_i$, for $i = 0, 1$, one considers a local feature $p_i(x) \in \mathbb{R}^s$, that is a low dimensional vector, intended to represent the geometry of Ω_i around x. To perform a recognition that is invariant to a class of deformations $\mathcal{R}$, one needs to build features that are invariant under $\mathcal{R}$. It means that if $\Omega_0 = R(\Omega_1)$ for $R \in \mathcal{R}$, then

$$\forall\, t \in [0,1], \quad p_0(\gamma_0(t)) = p_1(\gamma_1(\varphi(t))).$$

The vector $p_i(x) \in \mathbb{R}^s$ usually gives a multi-scale local representation of Ω_i around x, and makes use of s different resolutions. A popular choice is the curvature of γ_i at various scales $\{\sigma_j\}_{j=0}^{s-1}$

$$\forall\, 0 \leqslant j < s, \quad (p_i(x))_j = \kappa(\gamma_i \star G_{\sigma_j}, t), \tag{5.18}$$

where $\gamma_i \star G_{\sigma_j}$ denotes the component-by-component convolution of the curve with a Gaussian kernel of variance σ^2, and κ is the curvature as defined in (3.5)

$$n(\gamma,t) = \frac{\gamma'(t)^\perp}{\|\gamma'(t)\|}, \quad \text{and} \quad \kappa(\gamma,t) = \langle n'(\gamma,t),\, \gamma'(t)\rangle \frac{1}{\|\gamma'(t)\|^2}.$$

Using a continuous set of scales defines a curvature scale space [193]. In practice, scales are often sampled according $\sigma_j = \sigma_0 a^j$ for some $a > 1$.

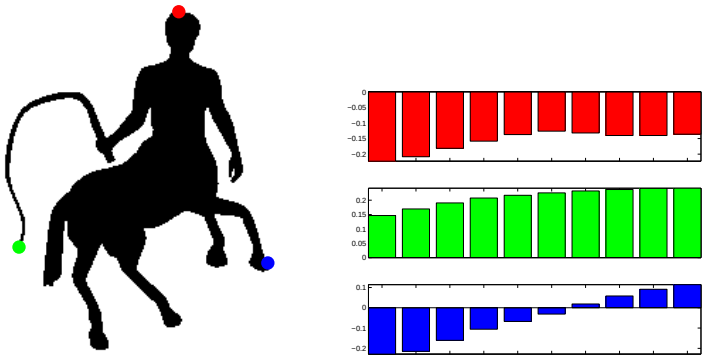
An alternative local set of features is computed by local integration over the shape domain [180]. For instance, one can use an averaging of the indicator function of the shape

$$\forall\, 0 \leqslant j < s, \quad (p_i(x))_j = (G_{\sigma_j} \star 1_{\Omega_i})(x) \tag{5.19}$$

where

$$1_\Omega(x) = \begin{cases} 1 & \text{if} \quad x \in \Omega, \\ 0 & \text{otherwise} \end{cases}$$

see [180] for the connection between differential features such as (5.18) and integral features such as (5.19).



Sampling locations x_i Descriptors p_{x_i}

Fig. 5.7 *Examples of multi-scale curvature descriptors p_{x_i} as defined in* (5.18) *for three different locations $x_i \in \partial\Omega$.*

Global geodesic features. In order to be invariant to non-rigid bending $R \in \mathcal{R}$, that is nearly isometric with respect to the geodesic structure of the shape

$$d_{\Omega_i}(x, y) \approx d_{R(\Omega_i)}(R(x), R(y)),$$

one needs to use more global features based on geodesic distances. Indeed, complicated bendings might change significantly local curvature indicators such as (5.18).

This can be achieved by defining $p_i(x)$ as the histogram of the geodesic distance $\{d_{\Omega_i}(x, y)\}_y$ to the point x, see (5.34).

Variational matching. For each $\eta \in [0,1]$, an optimal matching $\gamma_\eta : [0,1] \to [\eta, \eta+1]$ minimizes a weighted length

$$\min_{\gamma(0)=\eta, \gamma(1)=\eta+1} L(\gamma) = \int_0^1 W(t,\gamma(t))\sqrt{1+|\gamma'(t)|^2}\mathrm{d}t, \tag{5.20}$$

where the optimization should be restricted to strictly increasing mappings. The weight W takes into account the matching between the features

$$W(t,s) = \rho(\|p_0(t) - p_1(s)\|) > 0$$

where ρ is an increasing function. In this way, the optimal matching tries to link together parameter t and $s = \gamma(t)$ having similar features $p_0(t)$ and $p_1(s)$.

The final match $\gamma^\star$ between the parameters is the shortest match

$$\gamma^\star = \operatorname*{argmin}_{\eta \in [0,1]} L(\gamma_\eta). \tag{5.21}$$

Dynamic programming for matching. The minimization of a discretized version of the energy (5.20) can be performed using dynamic programming, see for instance [180, 159, 251, 274]. One discretizes $[0,1]^2$ on a square regular grid

$$\forall\, 0 \leqslant i,j < N, \quad (t_i, s_j) = (i/N, \eta + j/N) \tag{5.22}$$

of N^2 points. A directed graph is defined as

$$(i,j) \sim (i',j') \quad \Leftrightarrow \quad \left\{ \begin{array}{l} j' > j, \\ \mod(i'-j', N) < N/2 \\ \|i-j\| \leqslant \mu \end{array} \right. ,$$

where $\mod(i,N) \in \{0,\ldots,N-1\}$ is the usual modulo operator, and $\mu \geqslant 1$ controls the width of the connection. Increasing the value of μ and N makes the discrete optimization problem more precise.

A graph metric is then defined on each edge as

$$\forall\, e = ((i,j) \sim (i',j')), \quad W_e = (W(t_i,s_j) + W(t_{i'},s_{j'}))/2.$$

The discrete geodesic γ_η between the points $(0,0)$ and $(N-1,N-1)$ of the graph is obtained using the Dijkstra algorithm detailed in Section 2.2.3.

The search for the final match that solves (5.21) is performed by testing several values of η, and it can be accelerated by using heuristics.

Fast Marching for matching. As suggested in [121], one can relax the condition that the mapping γ is strictly increasing. This allows the use of the Fast Marching algorithm to compute a precise sub-pixel matching.

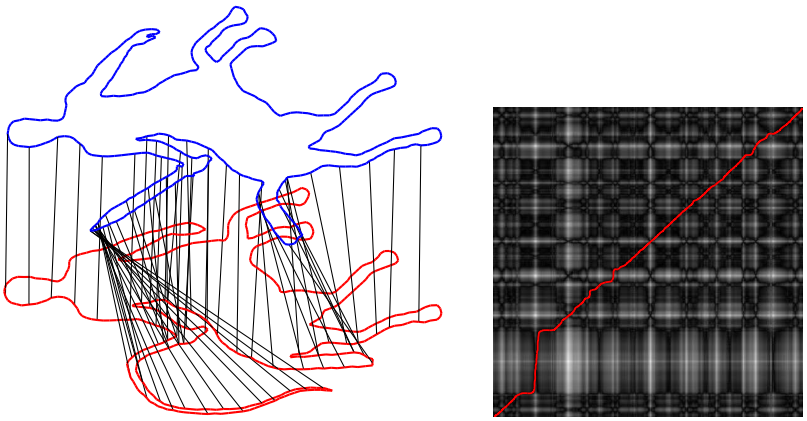
One replaces the variational problem (5.20) by a geodesic length minimization inside the square $\Omega_\eta = [0,1] \times [\eta, \eta+1]$

$$\min_{\gamma(0)=(0,\eta),\gamma(1)=(1,\eta+1)} L(\gamma) = \int_0^1 W(\gamma(r))\|\gamma'(r)\|\mathrm{d}r, \tag{5.23}$$

where $\gamma : [0,1] \to \Omega_\eta$ is a planar curve.

Finding the shortest curve γ_η that solves (5.23) is obtained by using the isotropic Fast Marching on the discrete grid (5.22), as detailed in Section 2.2.2, and then performing a gradient descent as detailed in Section 2.5.1.

Similarly to (5.21), the final matching $\gamma^\star(r) = (t^\star(r), s^\star(r))$ is the one that minimizes $L(\gamma_\eta)$ by varying η. The resulting matching is obtained by linking $t^\star(r) \leftrightarrow s^\star(r)$ for a varying r. One should note that this matching is not necessarily one to one.



Metric $W(t,s)$ and geodesic curve $\gamma^\star(r)$ Matching of the curves

Fig. 5.8 *Example of geodesic curve in* $[0,1]^2$ *and the corresponding matching of the curves.*

Figure 5.8 shows an example of matching between two curves obtained with this method.

5.3 Surface and Shape Retrieval Using Geodesic Descriptors

Content based 2D shape and 3D surface retrieval is an important problem in computer vision. It requires to design similarity measures Δ to discriminate shapes from different classes, while being invariant to certain deformations.

5.3.1 Feature-based Shape Retrieval

Computing correspondences between shapes, as detailed in Section 5.2, is computationally too intensive for fast retrieval applications. Fast similarity measures Δ are computed by extracting global or local features, and then performing some comparison between the features. An important goal in designing a similarity measure is to achieve invariance to some class $\mathcal{R}$ of deformations

$$\forall\, R \in \mathcal{R}, \quad \Delta(\Omega_0, \Omega_1) = \Delta(R(\Omega_0), R(\Omega_1)). \tag{5.24}$$

There is a large amount of literature on content-based retrieval using similarity measures. One should refer to the review papers on 2D shapes [281, 296] and 3D surfaces [52, 266] retrieval.

Global descriptors. Fast approaches to shape comparisons use a low dimensional manifold descriptor $\varphi(\Omega)$ that is usually a vector $\varphi(\Omega) \in \mathbb{R}^k$. To achieve invariance (5.24), one requires that the descriptors are invariant with respect to a family $\mathcal{R}$ of deformations

$$\forall\, R \in \mathcal{R}, \quad \varphi(R(\Omega)) = \varphi(\Omega). \tag{5.25}$$

A descriptor is a single point in a low dimensional space. It is usually faster to compute than the full embedding of the manifold $\varphi(\Omega) \subset \mathbb{R}^k$ using a dimensionality reduction method of Section 5.1.

Simple global features are computed using polynomial moments [267, 268, 171], or Fourier transform [295], see [232] for a review.

The spectrum of the Laplace Beltrami operator defines a descriptor invariant to rigid motion and to simple bendings [235]. Spectral dimensionality reduction [19]allows one to define spectral distances between manifolds that requires the computation of a few eigenvectors of the

Laplace-Beltrami operator, see for instance [186, 47, 48]. Shape distributions [206] compute descriptors as histogram of the distribution of Euclidean distance between points on the manifold. This is extended to bending invariant descriptors in [21, 137, 136] using geodesic distances. It is possible to replace the geodesic distance by a diffusion distance [126] computed by solving a linear Poisson PDE, which might be advantageous for some applications.

Similarities by matching local descriptors. Many other shape and surface representations do not make use of a single descriptor. They rather compute similarities by matching points of interest for which local descriptors are defined. Local shape contexts [20] are local 2D histogram of contours around points of interest. Geodesic shape context makes use of geodesic curves to gain bending invariance [174]. Local tomographic projection on tangent plane (spin images) [141] defines a set of local descriptors.

5.3.2 Similarity Measures

Once the descriptor map φ has been computed for the manifolds of interest, a similarity measure between manifolds is obtained by comparing the descriptors

$$\Delta(\Omega_0, \Omega_1) = \delta(\varphi(\Omega_0), \varphi(\Omega_1)),$$

where δ is a metric between vectors of $\mathbb{R}^k$. This ensures that the triangular inequality (5.14) is satisfied.

In the following, we detail only the most popular metrics. The choice of a particular metric δ depends on the targeted application, and on the specificities of the manifolds of interest. See [280] for a comparison of various metric for 2D shape comparison.

ℓ^p similarity measures. The most classical measures are the ℓ^p norms

$$\delta(a, b)^p = \sum_{i=0}^{k-1} |a_i - b_i|^p. \tag{5.26}$$

Kullback-Leiber divergence. A popular way to define a descriptor $\varphi(\Omega) \in \mathbb{R}^k$ is by computing a discrete histogram with k bins of some set of values that depend on the geometry of Ω. Such a histogram descriptor $a = \varphi_0(\Omega_0)$ satisfies the constraints

$$\forall\, i, \quad a_i \geqslant 0, \quad \text{and} \quad \sum_i a_i = 1.$$

Under these conditions, one can consider the Kullback-Leiber divergence,

$$\delta(a,b) = \sum_i a_i \log_2(a_i/b_i), \tag{5.27}$$

which is non-symmetric and does not satisfy the triangular inequality, so (5.14) does not hold. The resulting distance between shapes is however quite popular because of its simplicity.

Wassersein distance. Similarities (5.26) and (5.27) compare independently each entry of the descriptors. In particular, shuffling in the same manner the entries of both descriptors does not change the similarity. In order to take into account the position of the index in the descriptors, one can use the Wasserstein distance, also called the earth mover's distance [244] which is more complicated to compute. The ℓ^p Wasserstein distance is defined as a minimization among matrices $P \in \mathbb{R}^{k\times k}$

$$\delta(a,b)^p = \min_{P1=a, P^{\mathrm{T}}1=b, P\geqslant 0} \sum_{i,j} |i-j|^p P_{i,j}. \tag{5.28}$$

One can prove that $\delta(a,b)$ is a distance on probability distributions, see [282].

In this section, we consider only integer indexes (i,j), and the distance can be expressed using the inverse of the cumulative distribution

$$\delta(a,b)^p = \sum_i |C_a^{-1}(i) - C_b^{-1}(i)|^p \quad \text{where} \quad C_a(i) = \sum_{j\leqslant i} c_j \tag{5.29}$$

where C_a^{-1} is the inverse function of C_a. Some care is required to compute it if C_a is not strictly increasing, which is the case if $a_i = 0$ for some i.

The Wasserstein distance extends for indices i in any dimension (not necessarily integers), and computing (5.28) requires the resolution of a

linear program. It has been used in conjunction with geodesic distances to perform shape retrieval [**?**].

5.3.3 Geodesic Descriptors

One can design a family of global descriptors by considering the histogram of some functions defined on the manifold.

Euclidean shape distributions. Shape distributions [206] build global descriptors from a set of Euclidean distances

$$\forall\, 0 \leqslant i,j < N, \quad d_{i,j} = \|x_i - x_j\|, \tag{5.30}$$

where $\{x_i\}_i$ is a discrete uniform sampling of the manifold.

One builds from these distances real valued mappings, such as for instance the mean, median, maximum and minimum value of the distances to a fixed point, for all $0 \leqslant i < N$

$$f_i^{\min} = \min_{0 \leqslant j < N} d_{i,j}, \qquad f_i^{\max} = \max_{0 \leqslant j < N} d_{i,j}, \tag{5.31}$$

$$f_i^{\text{mean}} = \sum_{0 \leqslant j < N} d_{i,j}, \qquad f_i^{\text{median}} = \operatorname*{median}_{0 \leqslant j < N} d_{i,j}. \tag{5.32}$$

One can then define global descriptors by considering the histograms of these mappings

$$\varphi(\Omega) = \mathcal{H}(\{f_i^*\}_{0 \leqslant i < N}) \tag{5.33}$$

where $*$ is any of $\{\min, \max, \text{mean}, \text{median}\}$. The histogram $h = \mathcal{H}(Y) \in \mathbb{R}^k$ of a set of values $Y \subset \mathbb{R}$, assumed to be rescaled in $[0,1]$, is defined as

$$h_\ell = \frac{\text{card}(\{y \in Y \setminus \ell/k \leqslant y < (\ell+1)/k\})}{\text{card}(Y)}.$$

The resulting global descriptors $\varphi(\Omega)$ are invariant to rigid deformations, meaning that (5.25) holds for the set $\mathcal{R}$ of rigid motions.

Geodesic shape distributions. It is possible to extend these shape distribution descriptors (5.33) by replacing the Euclidean distance $\|\cdot\|$ by the geodesic distance d_Ω.

One can for instance consider the geodesic distance inside a planar shape $\Omega \subset \mathbb{R}^2$, inside a volumetric shape $\Omega \subset \mathbb{R}^3$, or on the boundary of a surface embedded in $\mathbb{R}^3$. They are all treated within the same geodesic framework. The planar and volumetric shapes correspond to the restriction of the identity metric $T_x = \mathrm{Id}_d$ to a sub-domain, as detailed in Section 1.3.2, while surfaces correspond to an anisotropic metric T_x, as detailed in Section 1.1.2.

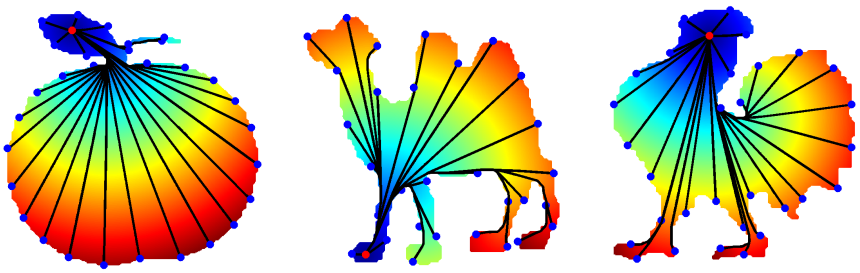


Fig. 5.9 *Geodesics inside a 2D shape.*

One thus replaces (5.30) by the following pairwise measures

$$d_{i,j} = d_\Omega(x_i, x_j),$$

where d_Ω is the length of the shortest path inside Ω linking x_i to x_j. Figure (5.9) shows some examples of such shortest paths.

The geodesic distance map $U_{x_i}(x) = d_\Omega(x_i, x)$ differs significantly from the Euclidean distance map $\|x_i - x\|$ when the shapes are non convex. Figure 5.10 shows an example of comparison.

The shape distribution has been extended to the geodesic setting on 3D meshes using the distribution of the mean f^{mean} [21] and to 2D shapes [137], volumetric shapes, and 3D surfaces [136] by considering the distribution of the maximum distance $f^{\max}$.

Figure 5.11 shows the examples of the maximum, minimum, mean and median geodesic distance to all the points within a given planar shape $\Omega \subset \mathbb{R}^2$.

The resulting global descriptors $\varphi(\Omega) = \mathcal{H}(f^*)$ are invariant to isometric deformations. More generally, $\varphi(\Omega_0) \approx \varphi(\Omega_1)$ for $\Omega_1 = R(\Omega_0)$

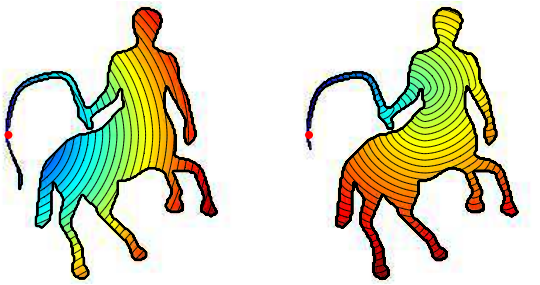


Fig. 5.10 *Comparison of Euclidean and geodesic distances inside a 2D shape (from the red point).*

if the deformation does not modify too much the geodesic distance, as measured for instance using $\delta(R)$ defined in (5.17). This is the case for bending deformation and articulation, see [174].

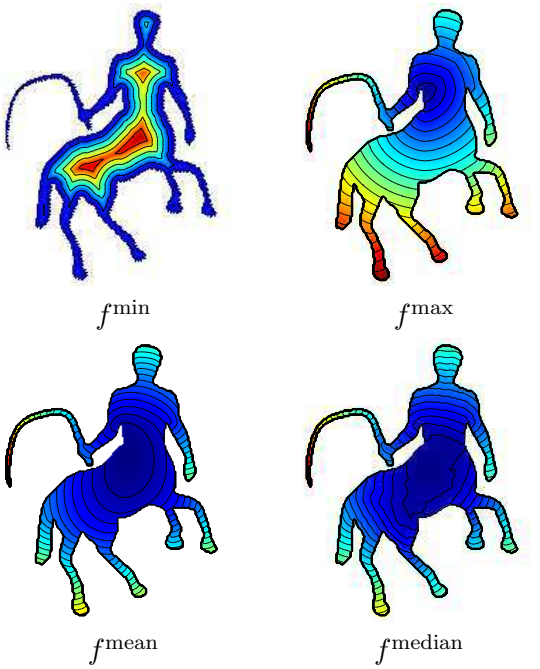


Fig. 5.11 *Example of several functions of the geodesic distances.*

Geodesic eccentricity. The maximum distance $f^{\max}$ is also known as the geodesic eccentricity of the manifold, see [137, 136]. For a continuous sampling of the manifold, it is defined as

$$f^{\max}(x) = \max_{y \in \Omega} d_\Omega(x, y).$$

This function has many interesting properties, in particular it can be computed from the distance to the boundary points

$$f^{\max}(x) = \max_{y \in \partial\Omega} d_\Omega(x, y),$$

which allows for a fast evaluation using a few fast marching propagation.

Figure 5.12 shows some examples of eccentricity function on shapes and surfaces.

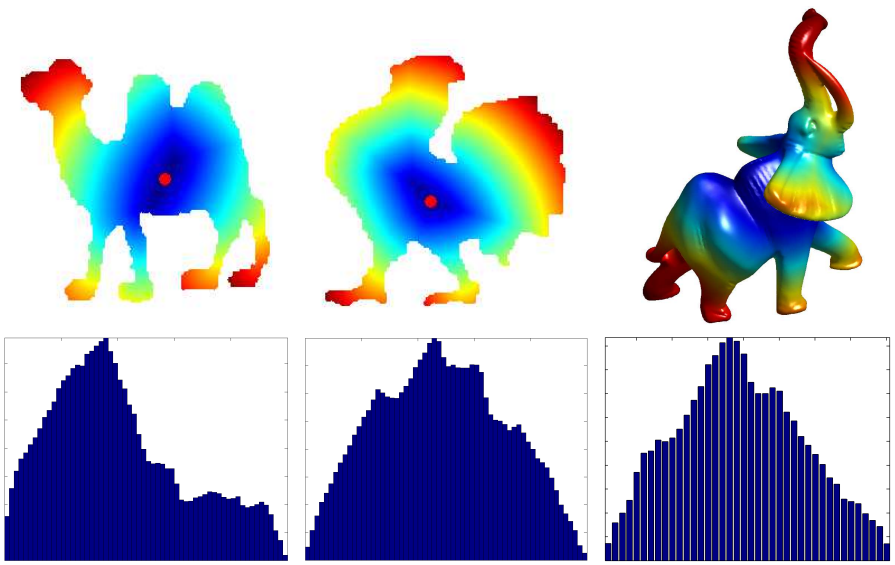


Fig. 5.12 *Top row: eccentricity function $f^{\max}$ on a planar shape (left and center, the red point corresponds to the minimum value) and on a 3D surface (right). Bottom row: histograms $\varphi(\Omega)$ corresponding to these eccentricity functions.*

Starting from a shape library $\{\Omega_1, \ldots, \Omega_p\}$, one can use the eccentricity shape descriptor $\varphi(\Omega)$ to do shape retrieval using for instance a

nearest neighbor classifier. More complex signatures can be constructed out of geodesic distances, and un-supervised recognition can also be considered. We refer to [137, 136] for a detailed study of the performance of shape recognition with eccentricity histograms. Figure 5.13 shows examples of typical shape retrievals using this approach.

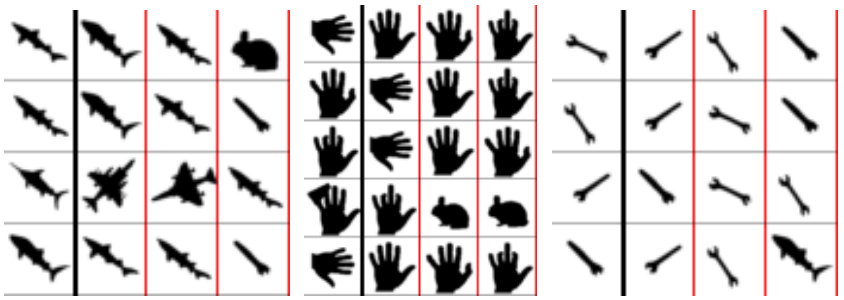


Fig. 5.13 *Example of shape retrieval using geodesic eccentricity, see [137] for details. The query shape is the leftmost shape of each row. The left part of the figure shows*

Local geodesic descriptors. Another way to describe these geodesic shape distributions is to use local descriptors p_x that are the histogram of the geodesic distance to x

$$p_x = \mathcal{H}(\{d_\Omega(x,y)\}_{y\in\Omega}) \in \mathbb{R}^k. \tag{5.34}$$

This descriptor is an indicator of the geometry of Ω seen from the point x.

One can see that the functions f^* defined in (5.31) corresponds to the application of some particular statistical estimators (maximum, minimum, mean or median values). This gives a recipe to build descriptors using statistical measures.

It can be used to compute a matching γ between the boundaries of two planar shapes, as a replacement for the differential or integral descriptors defined in (5.18) and (5.19).

Figure 5.14 shows an example of local geodesic descriptor for several locations in a planar shape.

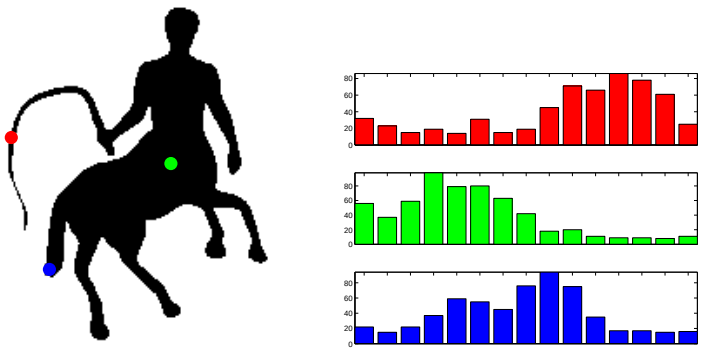


Fig. 5.14 *Examples of local geodesic descriptors* p_{x_i} *as defined in* (5.34) *for three different locations* $x_i \in \Omega$.

Conclusion

This monograph has reviewed fundamental and computational aspects of Riemannian manifolds, as well as applications in the fields of computer graphics and vision. Riemannian metrics bring together the concepts of spacial adaptivity, anisotropy and orientation within a mathematically sound formulation. They also offer fast computational algorithms that are suitable for large scale applications. These two important features work hand in hand to offer practical solutions to three important classes of shape and surface processing problems: segmentation, sampling and recognition.

Many exciting areas of research on geodesic methods are currently under investigation, or should deserve more attention. Faster algorithms, which offer good performances for highly anisotropic metrics, are desirable. Accelerating the computation of geodesic curves through efficient heuristics with theoretical guarantees is also relevant for many applications requiring real time performances. Deriving better geodesic sampling schemes with theoretical guarantees could improve the state of the art in surface remeshing and image compression. The problem of fast and accurate matching of shapes with bending invariance is mostly open, since solving high dimensional multi-dimensional scaling

is computationally too demanding for interactive retrieval applications. Finally, many other problems in vision and graphics that require handling datasets with strong anisotropy could certainly benefit from advances in geodesic methods.

References

[1] P. Agarwal and S. Suri. Surface Approximation and Geometric Partitions. *SIAM J. Comput.*, 19:1016–1035, 1998.

[2] J. C. Aguilar and J. B. Goodman. Anisotropic mesh refinement for finite element methods based on error reduction. *J. Comput. Appl. Math.*, 193(2):497–515, 2006.

[3] V. Akman. *Unobstructed shortest paths in polyhedral environments*. Springer-Verlag New York, Inc., New York, NY, USA, 1987.

[4] F. Alauzet. Size gradation control of anisotropic meshes. *Finite Elem. Anal. Des.*, 46:181–202, July 2010.

[5] P. Alliez, M. Attene, C. Gotsman, and G. Ucelli. Recent advances in remeshing of surfaces. In *Shape Analysis and Structuring*, pages 53–82. Springer, 2008.

[6] P. Alliez, D. Cohen-Steiner, O. Devillers, B. Lévy, and M. Desbrun. Anisotropic polygonal remeshing. *ACM Transactions on Graphics*, 22(3):485–493, 2003.

[7] P. Alliez, D. Cohen-Steiner, M. Y., and M. Desbrun. Variational tetrahedral meshing. *ACM Transactions on Graphics*, 24(3):617–625, July 2005.

[8] P. Alliez, E. Colin de Verdière, O. Devillers, and M. Isenburg. Isotropic surface remeshing. In *Proc. Shape Modeling International*, pages 49–58. IEEE Computer Society, 2003.

[9] N. Amenta, M. W. Bern, and D. Eppstein. The crust and the beta-skeleton: Combinatorial curve reconstruction. *Graphical Models and Image Processing*, 60(2):125–135, 1998.

[10] Pablo Arbelaez and Laurent D. Cohen. Energy partitions and image segmentation. *Journal of Mathematical Imaging and Vision*, 20(1-2):43–57, January - March 2004.

[11] Roberto Ardon and Laurent D. Cohen. Fast constrained surface extraction by minimal paths. *Int. J. Comput. Vision*, 69(1):127–136, 2006.

[12] Roberto Ardon, Laurent D. Cohen, and Anthony Yezzi. A new implicit method for surface segmentation by minimal paths in 3D images. *Applied Mathematics and Optimization*, 55(2):127–144, March 2007.

[13] N. Aspert, D. Santa-Cruz, and T. Ebrahimi. MESH: measuring error between surfaces using the hausdorff distance. *Proc. of IEEE International Conference on Multimedia and Expo 2002*, I:705–708, 2002.

[14] I. Babuška and A. K. Aziz. On the angle condition in the finite element method. *SIAM Journal on Numerical Analysis*, 13(2):214–226, April 1976.

[15] F. A. Baqai, J. H. Lee, A. U. Agar, and J. P. Allebach. Digital color halftoning. *IEEE Signal Processing Magazine*, 22(1):87–96, January 2005.

[16] P.J. Basser, J. Mattiello, and D. LeBihan. Estimation of the effective self-diffusion tensor from the nmr spin echo. *Journal of Magnetic Resonance B*, 103(3):247–254, 1994.

[17] P.J. Basser, J. Mattiello, and D. Lebihan. MR diffusion tensor spectroscopy and imaging. *Biophysical Journal*, 66:259–267, 1994.

[18] P. N. Belhumeur, D. J. Kriegman, and A. L. Yuille. The bas-relief ambiguity. *International Journal of Computer Vision*, 35(1):33–44, November 1999.

[19] M. Belkin and P. Niyogi. Laplacian eigenmaps for dimensionality reduction and data representation. *Neural Computation*, 15(6):1373–1396, 2003.

[20] S. Belongie, J. Malik, and J. Puzicha. Shape matching and object recognition using shape contexts. *IEEE Trans. Patt. Anal. and Mach. Intell.*, 24(4):509–522, 2002.

[21] A. Ben Hamza and H. Krim. Geodesic matching of triangulated surfaces. *IEEE Trans. Image Proc.*, 15(8):2249–2258, August 2006.

[22] F. Benmansour, G. Carlier, G. Peyré, and F. Santambrogio. Numerical approximation of continuous traffic congestion equilibria. *Networks and Heterogeneous Media*, 4(3):605–623, 2009.

[23] F. Benmansour and L. D. Cohen. Tubular structure segmentation based on minimal path method and anisotropic enhancement. *International Journal of Computer Vision*, to appear, 2010.

[24] Fethallah Benmansour and Laurent D. Cohen. Fast object segmentation by growing minimal paths from a single point on 2D or 3D images. *J. Math. Imaging Vis.*, 33(2):209–221, February 2009.

[25] M. Bern and D. Eppstein. Mesh Generation and Optimal Triangulation. In F. K. Hwang and D. Z. Du, editors, *Computing in Euclidean Geometry*. World Scientific, March 1992.

[26] M. Bernstein, V. de Silva, J.C. Langford, and J.B. Tenenbaum. Graph approximations to geodesics on embedded manifolds. *Standford Technical Report*, 2005.

[27] P.J. Besl and N.D. McKay. A method for registration of 3-d shapes. *IEEE Trans. Patt. Anal. and Mach. Intell.*, 14(2):239–256, 1992.

[28] S. Beucher. Watersheds of functions and picture segmentation. In *IEEE Int. Conf. on Acoustics, Speech and Signal Processing*, pages 1928–1931, Paris, 1982.

[29] T. N. Bishop, K. P. Bube, R. T. Cutler, R. T. Langan, P. L. Love, J. R. Resnick, R. T. Shuey, D. A. Spindler, and H. W. Wyld. Tomographic determination of velocity and depth in laterally varying media. *Geophysics*, 50(6):903–923, 1985.

[30] J-D. Boissonnat, C. Wormser, and M. Yvinec. Anisotropic diagrams: Labelle shewchuk approach revisited. *Theor. Comput. Sci*, 408(2-3):163–173, 2008.

[31] J-D. Boissonnat, C. Wormser, and M. Yvinec. Locally uniform anisotropic meshing. In *Proc. SCG'08*, pages 270–277, New York, NY, USA, 2008. ACM.

[32] I. Borg and P. Groenen. *Modern multidimensional scaling*. Springer-Verlag, New York, 1997. Theory and applications.

[33] F. Bornemann and C. Rasch. Finite-element discretization of static hamilton-jacobi equations based on a local variational principle. *Comput. Visual Sci.*, 9(2):57–69, 2006.

[34] H. Borouchaki, P.L. George, and B. Mohammadi. Delaunay mesh generation governed by metric specifications. Part I. algorithms. *Finite Elem. Anal. Des.*, 25(1-2):61–83, 1997.

[35] H. Borouchaki, F. Hecht, and P. J. Frey. Mesh gradation control. *Int. J. Numer. Meth. Engng*, 43(6):1143–1165, 1998.

[36] F. J. Bossen and P. S. Heckbert. A pliant method for anisotropic mesh generation. In *5th Intl. Meshing Roundtable*, pages 63–74, October 1996.

[37] S. Bougleux, G. Peyré, and L. Cohen. Image compression with geodesic anisotropic triangulations. *Proc. of ICCV'09*, 2009.

[38] S. Bougleux, G. Peyré, and L. D. Cohen. Anisotropic geodesics for perceptual grouping and domain meshing. In David A. Forsyth, Philip H. S. Torr, and Andrew Zisserman, editors, *Proc. of ECCV'08*, volume 5303 of *Lecture Notes in Computer Science*, pages 129–142. Springer, 2008.

[39] Y. Boykov and V. Kolmogorov. Computing geodesics and minimal surfaces via graph cuts. In *Proc. of ICCV'03*, pages 26–33. IEEE Computer Society, 2003.

[40] A. Bronstein, M. Bronstein, and R. Kimmel. *Numerical Geometry of Non-Rigid Shapes*. Springer, 2007.

[41] A. M. Bronstein, M. M. Bronstein, A. M. Bruckstein, and R. Kimmel. Analysis of two-dimensional non-rigid shapes. *International Journal of Computer Vision*, 78(1):67–88, June 2008.

[42] A. M. Bronstein, M. M. Bronstein, A. M. Bruckstein, and R. Kimmel. Partial similarity of objects, or how to compare a centaur to a horse. *International Journal of Computer Vision*, 84(2):163–183, 2009.

[43] A. M. Bronstein, M. M. Bronstein, and R. Kimmel. Efficient computation of isometry-invariant distances between surfaces. *SIAM J. Scientific Computing*, 28(5):1812–1836, 2006.

[44] A. M. Bronstein, M. M. Bronstein, and R. Kimmel. Calculus of nonrigid surfaces for geometry and texture manipulation. *IEEE Trans. Vis. Comput. Graph*, 13(5):902–913, 2007.

[45] A. M. Bronstein, M. M. Bronstein, and R. Kimmel. Expression-invariant representations of faces. *IEEE Trans. Image Proc.*, 16(1):188–197, January 2007.

[46] A. M. Bronstein, M. M. Bronstein, and R. Kimmel. Weighted distance maps computation on parametric three-dimensional manifolds. *Journal of Computational Physics*, 225(1):771–784, 2007.

[47] A. M. Bronstein, M. M. Bronstein, R. Kimmel, M. Mahmoudi, and G. Sapiro. A gromov-hausdorff framework with diffusion geometry for topologically-robust non-rigid shape matching. *International Journal of Computer Vision*, 89(3):266–286, 2010.

[48] A. M. Bronstein, M. M. Bronstein, M. Ovsjanikov, and L. J. Guibas. Shape google: geometric words and expressions for invariant shape retrieval. *ACM Transactions on Graphics*, 2010.

[49] M. M. Bronstein, A. M. Bronstein, and R. Kimmel. Generalized multidimensional scaling: a framework for isometry-invariant partial surface matching. *Proc. of the National Academy of Sciences*, 103(5):1168–1172, 2006.

[50] M. M. Bronstein, A. M. Bronstein, R. Kimmel, and I. Yavneh. Multigrid multidimensional scaling. *Numerical linear algebra with applications*, 13(2–3):149–171, 2006.

[51] D. Burago, Y. Burago, and S. Ivanov. *A Course in Metric Geometry*, volume 33. Springer-Verlag, 2001.

[52] B. Bustos, D. A. Keim, D. Saupe, T. Schreck, and D. V. Vranić. Feature-based similarity search in 3d object databases. *ACM Comput. Surv.*, 37(4):345–387, 2005.

[53] G.J. Butler. Simultaneous packing and covering in euclidean space. *Proc. London Math. Soc.*, 25:721–735, June 1972.

[54] G. Buttazzo, A. Davini, I. Fragal, and F. Maciá. Optimal riemannian distances preventing mass tranfer. *J. Reine Angew. Math.*, 575:157–171, 2004.

[55] J. Canny. A computational approach to edge detection. *IEEE Transactions Pattern Analysis and Machine Intelligence*, 8(6):679–698, November 1986.

[56] G. Carlier, C. Jimenez, and F. Santambrogio. Optimal transportation with traffic congestion and wardrop equilibria. *SIAM Journal on Control and Opt.*, 47(3):1330–1350, 2008.

[57] V. Caselles, F. Catté, T. Coll, and F. Dibos. A geometric model for active contours in image processing. *Numerische Mathematik*, 66(1):1–31, 1993.

[58] V. Caselles, R. Kimmel, and G. Sapiro. Geodesic active contours. *International Journal of Computer Vision*, 22(1):61–79, 1997.

[59] J. Chen and Y. Hahn. Shortest Path on a Polyhedron. *Proc. 6th ACM Sympos. Comput Geom*, pages 360–369, 1990.

[60] Y. Chen and G. Medioni. Object modelling by registration of multiple range images. *International Journal of Image and Vision Computing*, 10(3):145–155, April 1992.

[61] L. P. Chew. Constrained Delaunay triangulations. *Algorithmica*, 4:97–108, 1989.

[62] L. P. Chew. Guaranteed-quality triangular meshes. *Technical Report TR-89-983, Department of Computer Science, Cornell University*, 1989.

[63] L. P. Chew. Guaranteed-quality mesh generation for curved surfaces. In *Proc. of SCG '93*, pages 274–280, New York, NY, USA, 1993. ACM.

[64] D.L. Chopp. Some improvements of the fast marching method. *SIAM J. Sci. Comput.*, 23(1):230–244, 2001.

[65] P. Cignoni, C. Rocchini, and R. Scopigno. Metro: Measuring error on simplified surfaces. *Comput. Graph. Forum*, 17(2):167–174, 1998.

[66] U. Clarenz, M. Rumpf, and A. Telea. Robust feature detection and local classification for surfaces based on moment analysis. *IEEE Transactions on Visualization and Computer Graphics*, 10(5):516–524, 2004.

[67] K. L. Clarkson. Building triangulations using epsilon-nets. In Jon M. Kleinberg, editor, *Proc. of STOC*, pages 326–335. ACM, 2006.

[68] L. D. Cohen. Minimal paths and fast marching methods for image analysis. In *Handbook of Mathematical Methods in Computer Vision, N. Paragios and Y. Chen and O. Faugeras Editors, Springer*, 2005.

[69] L. D. Cohen and I. Cohen. Finite Element Methods for Active Contour Models and Balloons for 2-D and 3-D Images. *IEEE Trans. Patt. Anal. and Mach. Intell.*, 15, 1993.

[70] L. D. Cohen and R. Kimmel. Global Minimum for Active Contour models: A Minimal Path Approach. *International Journal of Computer Vision*, 24(1):57–78, Aug. 1997.

[71] Laurent Cohen and Ron Kimmel. Regularization properties for minimal geodesics of a potential energy. In *ICAOS*, 1996.

[72] Laurent D. Cohen. A new approach of vector quantization for image data compression and texture detection. In *International Conference on Pattern Recognition, ICPR'88*, Rome, 1988.

[73] Laurent D. Cohen and Thomas Deschamps. Grouping connected components using minimal path techniques. In *Proc. IEEE CVPR'01*, Kauai, Hawai, December 2001.

[74] Laurent D. Cohen and Thomas Deschamps. Multiple contour finding and perceptual grouping as a set of energy minimizing paths. In Springer, editor, *Proc. Third International Workshop on Energy Minimization Methods in Computer Vision and Pattern Recognition (EMMCVPR - 2001)*, 2001.

[75] Laurent D. Cohen and Thomas Deschamps. Segmentation of 3D tubular objects with adaptive front propagation and minimal tree extraction for 3D medical imaging. *Computer Methods in Biomechanics and Biomedical Engineering*, 10(4):289 – 305, August 2007.

[76] L.D. Cohen. On active contour models and balloons. *CVGIP: Image Underst.*, 53(2):211–218, 1991.

[77] L.D. Cohen. Multiple Contour Finding and Perceptual Grouping Using Minimal Paths. *J. Math. Imaging Vis.*, 14(3):225–236, May 2001.

[78] D. Cohen-Steiner and J-M. Morvan. Second fundamental measure of geometric sets and local approximation of curvatures. *J. Differential Geom.*, 74(3):363–394, 2006.

[79] J. H. Conway and N. J. A. Sloane. *Sphere Packings, Lattices and Groups*. Springer-Verlag, New York, NY, USA, 2nd edition, 1993.

[80] M. G. Crandall and P.-L. Lions. Two approximations of solutions of Hamilton–Jacobi equations. *Math. Comp.*, 43(167):1–19, 1984.

[81] M.G. Crandall, H. Ishii, and P.L. Lions. User's guide to viscosity solutions of second order partial differential equations. *Bulletin of the American Mathematical Society*, 27(1):1–67, 1992.

[82] M.G. Crandall and P.L. Lions. Viscosity solutions of hamilton-jacobi equations. *Transactions of the American Mathematical Society*, 277(1):1–42, 1983.

[83] O. Cuisenaire. *Distance Transformations: Fast Algorithms and Applications to Medical Image Processing*. PhD thesis, Université Catholique de Louvain, Louvain-La-Neuve, Belgique, 1999.

[84] O. Cuisenaire and B. Macq. Fast and exact signed Euclidean distance transformation with linear complexity. In *IEEE Intl Conference on Acoustics, Speech and Signal Processing (ICASSP99)*, Lecture Notes in Computer Science, pages 3293–3296. IEEE, 1999.

[85] O. Cuisenaire and B. Macq. Fast euclidean distance transformation by propagation using multiple neighborhoods. *Comput. Vis. Image Underst.*, 76(2):163–172, 1999.

[86] P.-E. Danielsson. Euclidean distance mapping. *Computer Graphics and Image Processing*, 14:227–248, 1980.

[87] S. Dasgupta and P. M. Long. Performance guarantees for hierarchical clustering. *J. Comput. Syst. Sci*, 70(4):555–569, 2005.

[88] M. de Berg, M. van Kreveld, M. Overmars, and O. Schwarzkopf. *Computational Geometry: Algorithms and Applications*. Springer-Verlag, second edition, 2000.

[89] J. de Leeuw. Applications of convex analysis to multidimensional scaling. *in Barra; Brodeau, F.; Romie, G. et al., Recent developments in statistics*, pages 133–145, 1977.

[90] V. de Silva and J.B. Tenenbaum. Sparse multidimensional scaling using landmark points. *Technical Report, Stanford University*, 2004.

[91] B. Delaunay. Sur la sphère vide. *Izvestia Akademii Nauk SSSR, Otdelenie Matematicheskikh i Estestvennykh Nauk*, 7:793–800, 1934.

[92] L. Demaret, N. Dyn, and A. Iske. Image compression by linear splines over adaptive triangulations. *Signal Processing*, 86(7):1604–1616, 2006.

[93] T. Deschamps and L.D. Cohen. Minimal paths in 3D images and application to virtual endoscopy. In *Proc. sixth European Conference on Computer Vision (ECCV'00)*, Dublin, Ireland, 26th June - 1st July 2000.

[94] T. Deschamps and L.D. Cohen. Fast extraction of minimal paths in 3D images and applications to virtual endoscopy. *Medical Image Analysis*, 5(4):281–299, December 2001.

[95] Thomas Deschamps and Laurent D. Cohen. Fast extraction of tubular and tree 3D surfaces with front propagation methods. In *Proc. 16th IEEE International Conference on Pattern Recognition (ICPR'02)*, pages 731–734, Quebec, Canada, August 2002.

[96] Thomas Deschamps and Laurent D. Cohen. Grouping connected components using minimal path techniques. In Springer, editor, *Geometrical Method in Biomedical image processing*. R. Malladi (ed.), 2002.

[97] A. Desolneux, L. Moisan, and J-M. Morel. A grouping principle and four applications. *IEEE Trans. Patt. Anal. and Mach. Intell.*, 25(4):508–513, 2003.

[98] Y. Devir, G. Rosman, A. M. Bronstein, M. M. Bronstein, and R. Kimmel. On reconstruction of non-rigid shapes with intrinsic regularization. *Proc. Workshop on Nonrigid Shape Analysis and Deformable Image Alignment (NORDIA)*, 2009.

[99] T. K. Dey. *Curve and Surface Reconstruction : Algorithms with Mathematical Analysis*. Cambridge University Press, 2007.

[100] T. K. Dey and P. Kumar. A simple provable algorithm for curve reconstruction. In *Proc. SODA'99*, pages 893–894, 1999.

[101] E. W. Dijkstra. A note on two problems in connexion with graphs. *Numerische Mathematik*, 1(1):269–271, December 1959.

[102] D. L. Donoho and C. Grimes. Hessian eigenmaps: Locally linear embedding techniques for high-dimensional data. *Proc. of the National Academy of Sciences*, 100(10):5591–5596, 2003.

[103] D. L. Donoho and C. Grimes. Image manifolds which are isometric to euclidean space. *J. Math. Imaging Vis.*, 23(1):5–24, July 2005.

[104] J. Doran. An approach to automatic problem-solving. *Machine Intelligence*, 1:105?–127, 1967.

[105] Q. Du and M. Emelianenko. Acceleration schemes for computing centroidal Voronoi tessellations. *Numerical linear algebra with applications*, 13(2–3):173–192, 2006.

[106] Q. Du, M. Emelianenko, and L. Ju. Convergence of the lloyd algorithm for computing centroidal voronoi tessellations. *SIAM Journal on Numerical Analysis*, 44:102–119, 2006.

[107] Q. Du, V. Faber, and M. Gunzburger. Centroidal Voronoi tessellations: Applications and algorithms. *SIAM Review*, 41:637–676, 1999.

[108] Q. Du and M. Gunzburger. Grid generation and optimization based on centroidal Voronoi tessellations. *Applied Mathematics and Computation*, 133(2–3):591–607, December 2002.

[109] Q. Du and D. Wang. Anisotropic centroidal Voronoi tessellations and their applications. *SIAM Journal on Scientific Computing*, 26(3):737–761, May 2005.

[110] N. Dyn, D. Levin, and S. Rippa. Data dependent triangulations for piecewise linear interpolation. *IMA J. Numer. Anal.*, 10(1):137–154, Jan. 1990.

[111] A. Elad (Elbaz) and R. Kimmel. On bending invariant signatures for surfaces. *IEEE Trans. Patt. Anal. and Mach. Intell.*, 25(10):1285–1295, 2003.

[112] Y. Eldar, M. Lindenbaum, M. Porat, and Y. Y. Zeevi. The farthest point strategy for progressive image sampling. *IEEE Trans. Image Proc.*, 6(9):1305–1315, September 1997.

[113] W.H. Press et Al. *Numerical Recipes in C : The Art of Computer Programming*. Cambridge University Press, 1988.

[114] R. Fabbri, L. Costa, J. Torelli, and O. Bruno. 2d euclidean distance transform algorithms: A comparative survey. *ACM Comput. Surv.*, 40(1):1–44, 2008.

[115] Z. Feng, I. Hotz, B. Hamann, and K. I. Joy. Anisotropic noise samples. *IEEE Transactions on Visualization and Computer Graphics*, 14(2):342–354, 2008.

[116] D.J. Field and A. Hayes R.F. Hess. Links contour integration by the human visual system: evidence for a local "association field". *Vision Res.*, 33(2):173–93, Jan 1993.

[117] M.S. Floater and K. Hormann. Surface parameterization: a tutorial and survey. In N. A. Dodgson, M. S. Floater, and M. A. Sabin, editors, *Advances in multiresolution for geometric modelling*, pages 157–186. Springer Verlag, 2005.

[118] R. W. Floyd and L. Steinberg. An adaptive algorithm for spatial greyscale. *Proceedings of the Society for Information Display*, 17(2):75–77, 1976.

[119] R.W. Floyd. Algorithm 245: Treesort. *Communications of the ACM*, 7(12):701, 1964.

[120] S. Fomel. A variational formulation of the fast marching eikonal solver. *Satnford Tech. Report 95, Stanford Exploration Project*, pages 127?–149, 1997.

[121] M. Frenkel and R. Basri. Curve matching using the fast marching method. In A. Rangarajan, editor, *proceedings of EMMCVPR*, pages 35–51, 2003.

[122] M. Garland and P. Heckbert. Surface simplification using quadric error metrics. *Proc. of SIGGRAPH 1997*, pages 209–215, 1997.

[123] P.L. George, H. Borouchaki, P.J. Frey, P. Laug, and E. Saltel. Mesh generation and mesh adaptivity: theory, techniques. *Encyclopedia of computational mechanics, E. Stein, R. de Borst and T.J.R. Hughes ed., John Wiley & Sons Ltd.*, 2004.

[124] A. V. Goldberg and C. Harrelson. Computing the shortest path: A search meets graph theory. In *Proc. of SODA*, pages 156–165. SIAM, 2005.

[125] T. F. Gonzalez. Clustering to minimize the maximum intercluster distance. *Theoretical Computer Science*, 38(2–3):293–306, June 1985.

[126] L. Gorelick, M. Galun, E. Sharon, R. Basri, and A. Brandt. Shape representation and classification using the poisson equation. *IEEE Trans. Patt. Anal. and Mach. Intell.*, 28(12):1991–2005, December 2006.

[127] M. Gromov. *Metric Structures for Riemannian and Non-Riemannian Spaces.* Birkhauser Boston, 1999.

[128] P. M. Gruber. Asymptotic estimates for best and stepwise approximation of convex bodies. I. *Forum Math.*, 5:281–297, 1993.

[129] P. M. Gruber. Optimum quantization and its applications. *Adv. Math.*, 186:456?497, 2004.

[130] G. Guy and G. Medioni. Inferring global perceptual contours from local features. *International Journal of Computer Vision*, 20(1-2):113–133, 1996.

[131] P. Hart, N. Nilsson, B., and Raphael. A formal basis for the heuristic determination of minimum-cost paths. *IEEE Trans. on Systems Science and Cybernetics*, SSC-4(2):100–107, July 1968.

[132] M.S. Hassouna and A.A. Farag. Robust skeletonization using the fast marching method. In *ICIP05*, pages I: 437–440, 2005.

[133] H. Hopf and W. Rinow. Ueber den begriff der vollstandigen differentialgeometrischen flachen. *Comm. Math. Helv.*, 3:209–225, 1931.

[134] H. Hoppe. Progressive meshes. *Proc. of SIGGRAPH 1996*, pages 99–108, 1996.

[135] B. K. P. Horn. Obtaining shape from shading information. In *The Psychology of Computer Vision*, pages 115–155, 1975.

[136] A. Ion, N. Artner, G. Peyré, S.B. López Mármol, W.G. Kropatsch, and L. Cohen. 3d shape matching by geodesic eccentricity. In *Proc. Workshop on Search in 3D*, Anchorage, Alaska, June 2008. IEEE.

[137] A. Ion, G. Peyré, Y. Haxhimusa, S. Peltier, W.G. Kropatsch, and L. Cohen. Shape matching using the geodesic eccentricity transform - a study. In C. Beleznai W. Ponweiser, M. Vincze, editor, *The 31st Annual Workshop of the Austrian Association for Pattern Recognition (OAGM/AAPR)*, pages 97–104, Schloss Krumbach, Austria, May 2007. OCG.

[138] A. K. Jain, M. N. Murty, and P. J. Flynn. Data clustering: a review. *ACM Comput. Surv.*, 31(3):264–323, 1999.

[139] S. Jbabdi, P. Bellec, R. Toro, J. Daunizeau, M. Pélégrini-Issac, and H. Benali. Accurate anisotropic fast marching for diffusion-based geodesic tractography. *Journal of Biomedical Imaging*, 2008(1):1–12, 2008.

[140] W-K. Jeong and R. T. Whitaker. A fast iterative method for eikonal equations. *SIAM J. Scientific Computing*, 30(5):2512–2534, 2008.

[141] A.E. Johnson and M. Hebert. Using spin images for efficient object recognition in cluttered 3d scenes. *IEEE Trans. Patt. Anal. and Mach. Intell.*, 21(5):433–449, 1999.

[142] H. Karcher. Riemannian center of mass and mollifier smoothing. *Commun. on Pure and Appl. Math.*, 5(30):509?–541, 1977.

[143] M. Kass, A. Witkin, and D. Terzopoulos. Snakes: Active contour models. *International Journal of Computer Vision*, 1(4):321–331, January 1988.

[144] Seongjai Kim. An $\wr(\backslash)$ level set method for eikonal equations. *SIAM Journal on Scientific Computing*, 22(6):2178–2193, 2001.

[145] B. B. Kimia, A. R. Tannenbaum, and S. W. Zucker. Shapes, shocks, and deformations I: the components of two-dimensional shape and the reaction-diffusion space. *International Journal of Computer Vision*, 15(3):189–224, 1995.

[146] R. Kimmel. *Numerical Geometry of Images: Theory, Algorithms, and Applications*. Springer, 2004.

[147] R. Kimmel, A. Amir, and A. M. Bruckstein. Finding shortest paths on surfaces using level sets propagation. *IEEE Trans. Patt. Anal. and Mach. Intell.*, 17(6):635–640, 1995.

[148] R. Kimmel, A. Amir, and A.M. Bruckstein. Finding shortest paths on surfaces using level sets propagation. *IEEE Trans. Patt. Anal. and Mach. Intell.*, 17(6):635–640, 1995.

[149] R. Kimmel and A. M. Bruckstein. Shape offsets via level sets. *Comp. Aid. Design*, 25:154–162, 1993.

[150] R. Kimmel and N. Kiryati. Finding the shortest paths on surfaces by fast global approximation and precise local refinement. *International Journal of Pattern Recognition and Artificial Intelligence*, 10(6):643–656, 1996.

[151] R. Kimmel and J. A. Sethian. Optimal algorithm for shape from shading and path planning. *J. Math. Imaging Vis.*, 14:2001, 2001.

[152] R. Kimmel and J.A. Sethian. Computing Geodesic Paths on Manifolds. *Proc. of the National Academy of Sciences*, 95(15):8431–8435, 1998.

[153] R. Kimmel and J.A. Sethian. Fast marching methods on triangulated domains. *Proc. of the National Academy of Sciences*, 95(15):8431–8435, July 1998.

[154] E. Klassen, A. Srivastava, W. Mio, and S. H. Joshi. Analysis of planar shapes using geodesic paths on shape spaces. *IEEE Trans. Patt. Anal. and Mach. Intell.*, 26(3):372–383, March 2004.

[155] E. Konukoglu. *Modeling Glioma Growth and Personalizing Growth Models in Medical Images*. PhD thesis, University of Nice-Sophia Antipolis, 2009.

[156] U. Kothe. Edge and junction detection with an improved structure tensor. In *Proc. DAGM03*, pages 25–32, 2003.

[157] J. B. Kruskal. Multidimensional scaling by optimizing goodness of fit to a nonmetric hypothesis. *Psychometrika*, 29:1–27, 1964.

[158] F. Labelle and J. R. Shewchuk. Anisotropic voronoi diagrams and guaranteed-quality anisotropic mesh generation. In *Proc. of SCG '03*, pages 191–200, New York, 2003. ACM Press.

[159] L. J. Latecki and R. Lakamper. Shape similarity measure based on correspondence of visual parts. *IEEE Trans. Patt. Anal. and Mach. Intell.*, 22(10):1185–1190, October 2000.

[160] J-C. Latombe. *Robot Motion Planning*. Kluwer Academic Publishers, Norwell, MA, USA, 1991.

[161] S. M. LaValle. *Planning Algorithms*. Cambridge University Press, Cambridge, U.K., 2006.

[162] H. Le. Locating Fréchet means with application to shape spaces. *Adv. Appl. Probab.*, 33:324–338, 2001.

[163] H. Le and D. G. Kendall. The riemannian structure of euclidean shape space: a novel environment for statistics. *Ann. Statist.*, 21:1225–1271, 1993.

[164] J. M. Lee. *Riemannian Manifolds*. Springer-Verlag, 1980.

[165] G. Leibon and D. Letscher. Delaunay triangulations and voronoi diagrams for riemannian manifolds. In *SCG '00: Proceedings of the sixteenth annual symposium on Computational geometry*, pages 341–349, New York, NY, USA, 2000. ACM.

[166] M. Levoy, K. Pulli, S. Rusinkiewicz, D. Koller, L. Pereira, M. Ginzton, S. Anderson, J. Davis, J. Ginsberg, B. Curless, J. Shade, and D. Fulk. The digital michelangelo project: 3D scanning of large statues. In *Proc. Siggraph 2000*, pages 131–144, New York, 2000. ACMPress.

[167] F. Leymarie and M. D. Levine. Simulating the grassfire transform using an active contour model. *IEEE Trans. Patt. Anal. and Mach. Intell.*, 14(1):56–75, January 1992.

[168] H. Li and A. Yezzi. Vessels as 4D curves: Global minimal 4D paths to extract 3D tubular surfaces and centerlines. *IEEE Trans. on Medical Imaging*, 26(9):1213–1223, 2007.

[169] H. Li, A. Yezzi, and Laurent D. Cohen. 3d multi-branch tubular surface and centerline extraction with 4d iterative key points. In *Proc. 12th International Conference on Medical Image Computing and Computer Assisted Intervention, MICCAI'09*, Imperial College, London, UK, 2009.

[170] X. Li, J-F. Remacle, N. Chevaugeon, and M. S. Shephard. Anisotropic mesh gradation control. In *Proc. of 13th Int. Meshing Rountable*, pages 401–412, 2004.

[171] S.X. Liao and M. Pawlak. On image analysis by moments. *IEEE Trans. Patt. Anal. and Mach. Intell.*, 18(3):254–266, 1996.

[172] S. Liapis, E. Sifakis, and G. Tziritas. Color and/or texture segmentation using deterministic relaxation and fast marching algorithms. In *ICPR*, pages Vol III: 617–620, 2000.

[173] M. Lin and D. Manocha. Collision and proximity queries. In *Handbook of Discrete and Computational Geometry.*, 2003.

[174] H. Ling and D. W. Jacobs. Shape classification using the inner-distance. *IEEE Trans. Patt. Anal. and Mach. Intell.*, 29(2):286–299, 2007.

[175] P. L. Lions, E. Rouy, and A. Tourin. Shape-from-shading, viscosity solutions and edges. *Numerische Mathematik*, 64(3):323–353, March 1993.

[176] Y. Liu, W. Wang, B. Lévy, F. Sun, D-M. Yan, L. Lu, and C. Yang. On centroidal voronoi tessellation - energy smoothness and fast computation. *ACM Transactions on Graphics*, 28(4), 2009.

[177] S. P. Lloyd. Least squares quantization in PCM. *IEEE Trans. Info. Theory*, 28(2):129–136, 1982.

[178] R. Malladi and J. A. Sethian. An $o(n \log(n))$ algorithm for shape modeling. *Proc. of the National Academy of Sciences*, 93:9389–9392, 1996.

[179] R. Malladi, J. A. Sethian, and B. C. Vemuri. Shape modeling with front propagation: A level set approach. *IEEE Trans. Patt. Anal. and Mach. Intell.*, 17(2):158–175, 1995.

[180] S. Manay, D. Cremers, B. W. Hong, A. J. Yezzi, and S. Soatto. Integral invariants for shape matching. *IEEE Trans. Patt. Anal. and Mach. Intell.*, 28(10):1602–1618, October 2006.

[181] P. Matula, J. Huben, and M. Kozubek. Fast marching 3d reconstruction of interphase chromosomes. In Milan Sonka, Ioannis A. Kakadiaris, and Jan Kybic, editors, *ECCV Workshops CVAMIA and MMBIA*, volume 3117 of *Lecture Notes in Computer Science*, pages 385–394. Springer, 2004.

[182] C. R. Maurer, R. Qi, and V. Raghavan. A linear time algorithm for computing exact Euclidean distance transforms of binary images in arbitrary dimensions. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 25(2):265–270, 2003.

[183] G. Medioni, M-S. Lee, and C-K. Tang. *A Computational Framework for Segmentation and Grouping*. Elsevier, 2000.

[184] A. Meijster, J.B.T.M. Roerdink, and W. H. Hesselink. A general algorithm for computing distance transforms in linear time. In *Mathematical Morphology and its Applications to Image and Signal Processing*, pages 331–340. Kluwer, 2000.

[185] F. Memoli. On the use of gromov-hausdorff distances for shape comparison. In *Proc. Symposium on Point Based Graphics 2007*, 2007.

[186] F. Mémoli. Spectral gromov-wasserstein distances for shape matching. In *Workshop on Non-Rigid Shape Analysis and Deformable Image Alignment (ICCV workshop, NORDIA'09)*, october 2009.

[187] F. Memoli and G. Sapiro. A theoretical and computational framework for isometry invariant recognition of point cloud data. *Found. Comput. Math.*, 5(3):313–347, 2005.

[188] F. Meyer. Topographic distance and watershed lines. *Signal Processing*, 38(1):113–125, July 1994.

[189] F. Meyer and P. Maragos. Multiscale morphological segmentations based on watershed, flooding, and eikonal PDE. In *Scale Space*, pages 351–362, 1999.

[190] J-M. Mirebeau and A. Cohen. Greedy bisection generates optimally adapted triangulations. Technical report, Laboratoire Jacques-Louis Lions, 2008.

[191] J. Mitchell, D. Mount, and C. Papadimitriou. The discrete geodesic problem. *SIAM Journal on Computing*, 16(4):647–668, 1987.

[192] J.S.B. Mitchell and C.H. Papadimitriou. The weighted region problem: finding shortest paths through a weighted planar subdivision. *J. ACM*, 38(1):18–73, 1991.

[193] F. Mokhtarian and A. K. Mackworth. A theory of multiscale, curvature-based shape representation for planar curves. *IEEE Trans. Patt. Anal. and Mach. Intell.*, 14(8):789–805, August 1992.

[194] U. Montanari. A method for obtaining skeletons using a quasi-euclidean distance. *J. ACM*, 15(4):600–624, 1968.

[195] D. Mumford. Mathematical theories of shape: Do they model perception? *Geometric Methods in Computer Vision*, 1570:2–10, 1991.

[196] D. Mumford. Elastica and computer vision. *In C. L. Bajaj (Ed.), Algebraic geometry and its applications*, pages 491–506, 1994.

[197] D. Mumford and J. Shah. Optimal approximations by piecewise smooth functions and associated variational problems. *Communications on Pure and Applied Mathematics*, XLII(4), 1989.

[198] L. Najman and M. Schmitt. Geodesic saliency of watershed contours and hierarchical segmentation. *IEEE Trans. Patt. Anal. and Mach. Intell.*, 18(12):1163–1173, 1996.

[199] M. Niemeijer, J.J. Staal, B. van Ginneken, M. Loog, and M.D. Abramoff. Comparative study of retinal vessel segmentation methods on a new publicly available database. In J. Michael Fitzpatrick and M. Sonka, editors, *SPIE Medical Imaging*, volume 5370, pages 648–656. SPIE, SPIE, 2004.

[200] B. Nilsson and A. Heyden. Segmentation of dense leukocyte clusters. In *Proc. of MMBIA '01*, page 221, Washington, DC, USA, 2001. IEEE Computer Society.

[201] N.J. Nilsson. *Problem-solving Methods in Artificial Intelligence*. McGraw-Hill, New York, 1971.

[202] M. Novotni and R. Klein. Computing geodesic distances on triangular meshes. In *Proc. WSCG'2002*, 2002.

[203] R.L. Ogniewicz. Discrete voronoi skeleton. *Hartung-Gorre*, 1993.

[204] Y. Ohtake, A. Belyaev, and H-P. Seidel. Ridge-valley lines on meshes via implicit surface fitting. *ACM Transactions on Graphics*, 23(3):609–612, 2004.

[205] K. Onishi and J. Itoh. Estimation of the necessary number of points in riemannian voronoi. In *Proc. of CCCG*, pages 19–24, 2003.

[206] R. Osada, T. Funkhouser, B. Chazelle, and D. Dobkin. Shape distributions. *ACM Transactions on Graphics*, 21(4):807–832, 2002.

[207] S. Osher and J. Sethian. Fronts propagating with curvature-dependent speed: Algorithms based on Hamilton-Jacobi formulations. *Journal of Computational Physics*, 79:12–49, 1988.

[208] S. J. Osher and R. Fedkiw. *Level Set Methods and Dynamic Implicit Surfaces*. Springer, 2002.

[209] S. J. Osher and N. Paragios. *Geometric Level Set Methods in Imaging, Vision, and Graphics*. Springer-Verlag, July 2003.

[210] V. Ostromoukhov. A simple and efficient error-diffusion algorithm. In *Proc. of SIGGRAPH*, pages 567–572, 2001.

[211] V. Ostromoukhov. Sampling with polyominoes. *ACM Transactions on Graphics*, 26(3):78–1–78–6, 2007.

[212] V. Ostromoukhov, C. Donohue, and P-M. Jodoin. Fast hierarchical importance sampling with blue noise properties. *ACM Transactions on Graphics*, 23(3):488–495, August 2004.

[213] N. Paragios, Y. Chen, and O. D. Faugeras. *Handbook of Mathematical Models in Computer Vision*. Springer, 2005.

[214] M. Péchaud. *Shortest paths calculations, and applications to medical imaging*. PhD thesis, Universit Paris 7/ENS/ENPC, September 2009.

[215] M. Péchaud, M. Descoteaux, and R. Keriven. Brain connectivity using geodesics in hardi. In *MICCAI*, London, England, September 2009.

[216] M. Péchaud, G. Peyré, and R. Keriven. Extraction of tubular structures over an orientation domain. In *CVPR '09: Proceedings of the 2009 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, Washington, DC, USA, 2009. IEEE Computer Society.

[217] X. Pennec. Intrinsic statistics on riemannian manifolds: Basic tools for geometric measurements. *J. Math. Imaging Vis.*, 25(1):127–154, 2006.

[218] G. Peyré and L. Cohen. Heuristically driven front propagation for fast geodesic extraction. *International Journal for Computational Vision and Biomechanics*, 1(1), 2007.

[219] G. Peyré and L. D. Cohen. Surface segmentation using geodesic centroidal tesselation. In *Proc. of 3DPVT*, pages 995–1002. IEEE Computer Society, 2004.

[220] G. Peyré and L. D. Cohen. Geodesic remeshing using front propagation. *International Journal of Computer Vision*, 69(1):145–156, 2006.

[221] Gabriel Peyré and Laurent D. Cohen. *Progress in Nonlinear Differential Equations and Their Applications*, volume 63, chapter Geodesic Computations for Fast and Accurate Surface Remeshing and Parameterization., pages 157–171. Birkhauser, 2005.

[222] S. Pippa and G. Caligiana. Gradh-correction: guaranteed sizing gradation in multi-patch parametric surface meshing. *Int. J. Numer. Meth. Engng*, 62(4):495–515, 2004.

[223] I. Pohl. Bi-directional search. In B. Meltzer and D. Michie, editors, *Machine Intelligence*, volume 6, pages 124–140. Edinburgh University Press, 1971.

[224] K. Polthier and M. Schmies. Straightest Geodesics on Polyhedral Surfaces. In H.-C. Hege and K. Polthier, editors, *Mathematical Visualization*, pages 135–150. Springer Verlag, 1998.

[225] K. Polthier and M. Schmies. Geodesic flow on polyhedral surfaces. In E. Gröller, H. Löffelmann, and W. Ribarsky, editors, *Data Visualization˜'99*, pages 179–188. Springer-Verlag Wien, 1999.

[226] Konrad Polthier and Markus Schmies. Straightest geodesics on polyhedral surfaces. In *SIGGRAPH '06: ACM SIGGRAPH 2006 Courses*, pages 30–38, New York, NY, USA, 2006. ACM.

[227] Miranda Poon, Ghassan Hamarneh, and Rafeef Abugharbieh. Live-vessel: Extending livewire for simultaneous extraction of optimal medial and boundary paths in vascular images. In *Proceedings of MICCAI (2)*, pages 444–451, 2007.

[228] A.M. Popovici and J.A. Sethian. 3-d imaging using higher order fast marching traveltimes. *Geophysics*, 67(2):604–609, 2007.

[229] H. Pottmann, J. Wallner, Q-X. Huang, and Y-L. Yang. Integral invariants for robust geometry processing. *Computer Aided Geometric Design*, 26(1):37–60, 2009.

[230] E. Prados, F. Camilli, and O. Faugeras. A unifying and rigorous shape from shading method adapted to realistic data and applications. *J. Math. Imaging Vis.*, 25(3):307–328, 2006.

[231] Emmanuel Prados, Christophe Lenglet, Jean-Philippe Pons, Nicolas Wotawa, Rachid Deriche, Olivier Faugeras, and Stefano Soatto. Control theory and fast marching techniques for brain connectivity mapping. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, New York, NY, June, 2006*, volume 1, pages 1076–1083. IEEE, June 2006.

[232] Richard J. Prokop and Anthony P. Reeves. A survey of moment-based techniques for unoccluded object representation and recognition. *CVGIP: Graphical Models and Image Processing*, 54(5):438–460, September 1992.

[233] J. Qian, Y-T. Zhang, and H-K. Zhao. Fast sweeping methods for eikonal equations on triangulated meshes. *SIAM Journal on Numerical Analysis*, 45:83–107, 2007.

[234] J.H. Reif and Z. Sun. Movement planning in the presence of flows. *Algorithmica*, 39(2):127–153, 2004.

[235] M. Reuter, F-E. Wolter, and N. Peinecke. Laplace-spectra as fingerprints for shape matching. In L. Kobbelt and V. Shapiro, editors, *Symposium on Solid and Physical Modeling*, pages 101–106. ACM, 2005.

[236] J. Rickett and S. Fomel. A second-order fast marching eikonal solver, January 13 2000.

[237] Shmuel Rippa. Long and thin triangles can be good for linear interpolation. *SIAM J. Numer. Anal.*, 29(1):257–270, Feb. 1992.

[238] A. Rosenfeld and J. L. Pfaltz. Distance functions on digital pictures. *Pattern Recognition*, 1(1):33–61, 1968.

[239] G. Rosman, A. M. Bronstein, M. M. Bronstein, A. Sidi, and R. Kimmel. Fast multidimensional scaling using vector extrapolation. *Techn. Report CIS-2008-01, Dept. of Computer Science, Technion, Israel*, 2008.

[240] G. Rosman, M. M. Bronstein, A. M. Bronstein, and R. Kimmel. Nonlinear dimensionality reduction by topologically constrained isometric embedding. *International Journal of Computer Vision*, 8(1):56–68, 2010.

[241] J. R. Rossignac and A. A. G. Requicha. Offsetting operations in solid modeling. *Comp. Aid. Design*, 3(2):129–148, August 1986.

[242] E. Rouy and A. Tourin. A viscosity solutions approach to shape-from-shading. *SIAM Journal on Numerical Analysis*, 29(3):867–884, 1992.

[243] S. T. Roweis and L. K. Saul. Nonlinear dimensionality reduction by locally linear embedding. *Science*, 290(5500):2323–2326, December 2000.

[244] Y. Rubner, C. Tomasi, and L. J. Guibas. The earth mover's distance as a metric for image retrieval. *International Journal of Computer Vision*, 40(2):99–121, November 2000.

[245] J. Ruppert. A Delaunay refinement algorithm for quality 2-dimensional mesh generation. *J. Algorithms*, 18(3):548–585, 1995.

[246] T. Saito and J. I. Toriwaki. New algorithms for Euclidean distance transformation of an n-dimensional digitized picture with applications. *Pattern Recognition*, 27(11):1551–1565, 1994.

[247] S.Beucher and C. Lantuejoul. Geodesic distance and image analysis. In *5th International Congress for Stereology, Salzburg, Austria*, pages 138–142, 1979.

[248] S.Beucher and C. Lantuejoul. On the use of the geodesic metric in image analysis. *Journal of Microscopy*, 121:39–49, January 1981.

[249] Alexander Schrijver. *Combinatorial optimization : polyhedra and efficiency.* volume A, paths, flows, matchings, chapter 1-38. Springer, 2003. Schrijver.

[250] E.L. Schwartz, A. Shaw, and E. Wolfson. A numerical solution to the generalized mapmaker's problem: Flattening nonconvex polyhedral surfaces. *IEEE Trans. Patt. Anal. and Mach. Intell.*, 11(9):1005–1008, 1989.

[251] T. B. Sebastian, P. N. Klein, and B. B. Kimia. On aligning curves. *IEEE Trans. Patt. Anal. and Mach. Intell.*, 25(1):116–124, January 2003.

[252] T. B. Sebastian, P. N. Klein, and B. B. Kimia. Recognition of shapes by editing their shock graphs. *IEEE Trans. Patt. Anal. and Mach. Intell.*, 26(5):550–571, May 2004.

[253] Jean Serra. *Image Analysis and Mathematical Morphology.* Academic Press, Inc., Orlando, FL, USA, 1983.

[254] J. A. Sethian. A fast marching level set method for monotonically advancing fronts. *Proc. of the National Academy of Sciences*, 93(4):1591–1595, 1996.

[255] J.A. Sethian. *Level Sets Methods and Fast Marching Methods.* Cambridge University Press, 2nd edition, 1999.

[256] J.A. Sethian and A. Vladimirsky. Ordered upwind methods for static hamilton–jacobi equations: Theory and algorithms. *SIAM Journal on Numerical Analysis*, 41(1):325–363, 2003.

[257] A. Sheffer, E. Praun, and K. Rose. Mesh parameterization methods and their applications. *Foundations and Trends in Computer Graphics and Vision*, 2(2), 2006.

[258] J. R. Shewchuk. Delaunay refinement algorithms for triangular mesh generation. *Comput. Geom*, 22(1-3):21–74, 2002.

[259] J. R. Shewchuk. What is a good linear element? interpolation, conditioning, and quality measures. In *International Meshing Roundtable*, pages 115–126, 2002.

[260] F.Y. Shih and Y.T. Wu. The efficient algorithms for achieving euclidean distance transformation. *IEEE Trans. Image Proc.*, 13(8):1078–1091, August 2004.

[261] F.Y. Shih and Y.T. Wu. Fast euclidean distance transformation in two scans using a 3x3 neighborhood. *Computer Vision and Image Understanding*, 93(2):195–205, February 2004.

[262] E. Sifakis, C. Garcia, and G. Tziritas. Bayesian level sets for image segmentation. *Journal of Visual Communication and Image Representation*, 13(1/2):44–64, March 2002.

[263] A. Spira and R. Kimmel. An efficient solution to the eikonal equation. *Interfaces and Free Boundaries*, 6(3):315–327, 2004.

[264] V. Surazhsky, T. Surazhsky, D. Kirsanov, S.J. Gortler, and H. Hoppe. Fast exact and approximate geodesics on meshes. *ACM Transactions on Graphics*, 24(3):553–560, 2005.

[265] M. Tang, M. Lee, and Y.J. Kim. Interactive hausdorff distance computation for general polygonal models. In *Proc. SIGGRAPH '09*, pages 1–9, New York, NY, USA, 2009. ACM.

[266] J. W. H. Tangelder and R. C. Veltkamp. A survey of content based 3D shape retrieval methods. *Multimedia Tools Appl*, 39(3):441–471, 2008.

[267] M. Teague. Image analysis via the general theory of moments. *Journal of the Optical Society of America*, 70(8):920–930, August 1980.

[268] C. H. Teh and R. T. Chin. On image analysis by the methods of moments. *IEEE Trans. Patt. Anal. and Mach. Intell.*, 10(4):496–513, July 1988.

[269] A. Telea and J. J. van Wijk. An augmented fast marching method for computing skeletons and centerlines. In D. Ebert, P. Brunet, and I. Navazo, editors, *Proceedings of the symposium on Data visualisation 2002*, pages 251–259, Barcelona, Spain, 2002. Eurographics Association.

[270] J. B. Tenenbaum, V. de Silva, and J. C. Langford. A Global Geometric Framework for Nonlinear Dimensionality Reduction. *Science*, 290(5500):2319–2323, Dec. 2000.

[271] N. Thorstensen and R. Keriven. Non-rigid shape matching using geometry and photometry. In *Proceedings ACCV*, 2009.

[272] L. Torresani, V. Kolmogorov, and C. Rother. Feature correspondence via graph matching: Models and global optimization. In *Proc. ECCV*, pages 596?–609, 2008.

[273] A. Treuille, S. Cooper, and Z. Popović. Continuum crowds. *ACM Transactions on Graphics*, 25(3):1160–1168, 2006.

[274] A. Trouve and L. Younes. Diffeomorphic matching problems in one dimension: Designing and minimizing matching functionals. In *Proc. of ECCV*, pages I: 573–587, 2000.

[275] Y-H. R. Tsai, L-T. Cheng, S. Osher, and H-K. Zhao. Fast sweeping algorithms for a class of Hamilton–Jacobi equations. *SIAM Journal on Numerical Analysis*, 41(2):673–694, April 2003.

[276] J. Tsitsiklis. Efficient Algorithms for Globally Optimal Trajectories. *IEEE Trans. on Automatic Control*, 40(9):1528–1538, Step. 1995.

[277] W.T. Tutte. How to draw a graph. *Proc. London Math*, 13:743–768, 1963.

[278] R. Van Uitert and I. Bitter. Subvoxel precise skeletons of volumetric data based on fast marching methods. *Medical physics*, 34:627–638, 2007.

[279] S. Valette, J.M. Chassery, and R. Prost. Generic remeshing of 3d triangular meshes with metric-dependent discrete Voronoi diagrams. *IEEE Trans Visu Comp Graph*, 14(2):369–381, 2008.

[280] R. C. Veltkamp. Shape matching: Similarity measure and algorithms. In *Proceedings Shape Modelling International*, pages 188–197. IEEE Press, 2001.

[281] R. C. Veltkamp and L.J. Latecki. Properties and performance of shape similarity measures. In *Proceedings of the 10th IFCS Conference on Data Science and Classification*, Slovenia, July 2006.

[282] C. Villani. *Topics in Optimal Transportation*. American Mathematical Society, 2003.

[283] L. Vincent and P. Soille. Watersheds in digital spaces: An efficient algorithm based on immersion simulations. *IEEE Trans. Patt. Anal. and Mach. Intell.*, 13(6):583–598, 1991.

[284] D. Wagner and T. Willhalm. Geometric speed-up techniques for finding shortest paths in large sparse graphs. In Giuseppe Di Battista and Uri Zwick, editors, *Proc. ESA*, volume 2832, pages 776–787. Springer, 2003.

[285] C. Wang, M. M. Bronstein, and N. Paragios. Discrete minimum distortion correspondence problems for non-rigid shape matching. *INRIA Report*, 7333, 2010.

[286] J.G. Wardrop. Some theoretical aspects of road traffic research. *Proc. Inst. Civ. Eng.*, 2(2):325–378, 1952.

[287] O. Weber, Y. S. Devir, A. M. Bronstein, M. M. Bronstein, and R. Kimmel. Parallel algorithms for approximation of distance maps on parametric surfaces. *ACM Transactions on Graphics*, 27(4), 2008.

[288] J.W. J. Williams. Algorithm 232: Heapsort. *Communications of the ACM*, 7:347–348, 1964.

[289] L.R. Williams and D. W. Jacobs. Stochastic completion fields: a neural model of illusory contour shape and salience. *Neural Comput.*, 9(4):837–858, 1997.

[290] E. Wolfson and E.L.Schwartz. Computing minimal distances on polyhedral surfaces. *IEEE Trans. Patt. Anal. and Mach. Intell.*, 11(9):1001–1005, 1989.

[291] R. P. Woods. Characterizing volume and surface deformations in an atlas framework: theory, applications, and implementation. *NeuroImage*, 18(3):769–788, 2003.

[292] S. Yamakawa and K. Shimada. High quality anisotropic tetrahedral mesh generation via ellipsoidal bubble packing. In *9th International Meshing Roundtable*, pages 263–274, 2000.

[293] L. Yatziv, A. Bartesaghi, and G. Sapiro. O(n) implementation of the fast marching algorithm. *J. Comput. Phys.*, 212(2):393–399, 2006.

[294] Y. Yokosuka and K. Imai. Guaranteed-quality anisotropic mesh generation for domains with curves. In *Proc. of EWCG'06*, 2006.

[295] C. T. Zahn and R. Z. Roskies. Fourier descriptors for plane closed curves. *IEEE Transactions on Computer*, 21(3):269–281, March 1972.

[296] D. S. Zhang and G. J. Lu. Review of shape representation and description techniques. *Pattern Recognition*, 37(1):1–19, January 2004.

[297] R. Zhang, P-S. Tsai, J.E. Cryer, and M. Shah. Shape from shading: A survey. *IEEE Trans. Patt. Anal. and Mach. Intell.*, 21(8):690–706, 1999.

[298] H. Zhao. Parallel implementations of the fast sweeping method. *Journal of Computational Mathematics*, 25(4):421–429, 2007.

[299] L. Zhou, M. S. Rzeszotarski, L. J. Singerman, and J. M. Chokreff. The detection and quantification of retinopathy using digital angiograms. *IEEE Trans. on Medical Imaging*, 13(4):619–626, 1994.

[300] S. C. Zhu. Stochastic jump-diffusion process for computing medial axes in markov random fields. *IEEE Trans. Patt. Anal. and Mach. Intell.*, 21(11):1158–1169, November 1999.

[301] S. C. Zhu and A. L. Yuille. FORMS: A flexible object recognition and modelling system. *International Journal of Computer Vision*, 20(3):187–212, 1996.

[302] G. Zigelman, R. Kimmel, and N. Kiryati. Texture Mapping Using Surface Flattening via Multi-dimensional Scaling. *IEEE Trans. on Visualization and Computer Graphics*, 8(1):198–207, 2002.